VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra měřicí a řídicí techniky

Vzdálený řídicí systém mobilního robotického zařízení

Remote Control System for Mobile Robotic Device

Ostrava, 2010 **Bc. Tomáš Lippa**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra měřicí a řídicí techniky

Zadání diplomové práce

Student: **Bc. Tomáš Lippa**

Studijní program: N2649 Elektrotechnika

Studijní obor: 2601T004 Měřicí a řídicí technika

Téma: Vzdálený řídicí systém mobilního robotického zařízení
Remote Control System for Mobile Robotic Device

Zásady pro vypracování:

1. Návrh a realizace ovládacího a pozorovacího panelu.
2. Výběr vhodného způsobu komunikace (volné pásma 2.4 GHz, Wifi, GSM ,..).
3. Platforma ARM Freescale iMX31 LiteKit.
4. Výběr vhodného OS (Linux, QNX, Win CE, ...).
5. Zobrazení průběhů a uložení naměřených hodnot.
6. Zhodnocení dosažených výsledků.

Seznam doporučené odborné literatury:

1. Timesys - Embedded Linux software [online]. c2009 [cit. 2009-11-11]. Dostupný z WWW.
2. Logic : A Product Development and Manufacturing Company [online]. c2009 [cit. 2009-11-11]. Dostupný z WWW.
3. Freescale Semiconductors [online]. c2009 [cit. 2009-11-11]. Dostupný z WWW: .
4. Qt : Develop applications and user interfaces once [online]. c2009 [cit. 2009-11-11]. Dostupný z WWW.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Jiří Kotzian, Ph.D.**

Datum zadání: 20.11.2009

Datum odevzdání: 07.05.2010

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky * OSTRAVA *

doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. Ing. Ivo Vondrák, CSc.
děkan fakulty

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Bc. Tomáš Lippa

Datum odevzdání diplomové práce: 7. 5. 2010

Poděkování

Chtěl bych tímto způsobem poděkovat vedoucímu mé diplomové práce panu Ing. Jiřímu Kotzianovi, Ph.D. za cenné rady a konzultace spojené s vypracováním mého úkolu.

Také bych chtěl poděkovat Ing. Viktoru Michnovi a Ing. Vilému Srovnalovi za rady spojené s vývojem grafické aplikace a s knihovnou Qt.

Poslední dík patří kolegům z týmu, Bc. Hynku Prokopovi, Bc. Marianu Kurucovi a Bc. Jaromíru Konečnému, kteří se semnou společně podíleli na projektu průzkumného vozidla.

Abstrakt

Diplomová práce se zabývá návrhem a realizací vzdáleného řídicího systému pro mobilní robotnické zařízení (průzkumné vozidlo) na platformě ARM Freescale i.MX31 LiteKit. Práce popisuje návrh aplikace s graficko-uživatelským rozhraním v operačním systému Timesys Linux, volbu vhodné grafické knihovny a výběr zařízení pro uživatelský přístup.

Řídicí systém zpracovává údaje získané z průzkumného vozidla, zobrazuje je na displeji a umožňuje také funkci manuálního řízení. Komunikuje s průzkumným vozidlem prostřednictvím bezdrátové sítě.

Klíčová slova

Timesys Linux, TimeStorm, Qt Creator, grafická knihovna Qt, ovládací panel, displej, joystick, i.MX31, WiFi.

Abstract

This thesis deals with the design and realization of the remote control system for mobile robotic device (exploratory vehicle) with platform ARM Freescale i.MX31 LiteKit. This work describes the design of application with graphical user interface on Timesys Linux operation system, choice of suitable graphic library and device for user access.

The control system processes data from exploratory vehicle, shows them on display and also enable function of manual control. The system communicates with exploratory vehicle through wireless net.

Keywords

Timesys Linux, TimeStorm, Qt Creator, graphical library Qt, control panel, display, joystick, i.MX31, WiFi.

Seznam použitých symbolů a zkratek

API	-	(Application Programming Interface) Rozhraní pro programování aplikací
ATA	-	(Advanced Technology Attachment) Standardní počítačová sběrnice
Bluetooth	-	Bezdrátová komunikační technologie
C	-	Programovací jazyk
C++	-	Programovací jazyk
CAN	-	(Controller Area Network), lze volně přeložit jako datová sběrnice místní sítě
CMOS	-	(Complemetary Metal-Oxide-Semicondudtor) Technologie integrovaných obvodů
CRC	-	(Cyclic Redundancy Check) Cyklický redundantní součet
Embedded	-	Vestavěný systém
Ethernet	-	Komunikační sběrnice
Flash	-	Elektronicky programovatelná paměť s náhodným přístupem
GSM	-	(Globalní Systém pro Mobilní komunikaci) Standard pro mobilní telefony
GPS	-	(Global Positioning System) Globální polohový systém
GUI	-	(Graphical User Interface) Graficko-uživatelské rozhraní
HDD	-	(Hard Disk Drive) Pevný disk, zařízení k uchování většího množství dat
HW	-	(Hardware) Technické vybavení počítače
IDE	-	(Integrated Development Environment) Softwarová aplikace s nástroji pro tvorbu programů
i.MX31	-	Vývojová deska na platformě ARM s procesorem Freescale
IEEE	-	(Institute of Electrical and Electronics Engineers) Standard technologie WiFi
Kernel	-	Jádro operačního systému
LCD	-	(Liquid Crystal Display) Zobrazovací zařízení
Linux	-	Operační systém
Mac	-	Operační systém
NAND	-	Typ paměti s blokovým zápisem
NFS	-	(Network File Systém) Internetový protokol pro vzdálený přístup k souborům
NOR	-	Typ paměti s bajtovým zápisem
PC	-	(Personal Computer) Osobní počítač
QNX	-	Operační systém reálného času
Qt	-	Knihovna pro vytváření grafických uživatelských rozhraní v jazyce C++
Qt Creator	-	Vývojové prostředí pro grafické aplikace s Qt knihovnou
RAM	-	(Random Access Memory) paměť s náhodným přístupem
RFS	-	Root File System
RS232	-	Standard pro komunikační sériovou linku
SD	-	(Secure Digital) Paměťová karta
SDL	-	(Simple DirectMedia Layer) Multimediální knihovna s nízko-úrovňovým přístupem
SDRAM	-	(Synchronous Dynamic Random Access Memory) Paměť se synchronním přenosem dat
SIM	-	(Subscripter Identify Module) Identifikační karta s mobilní síti

SPI	-	(Serial Peripheral Interface) Sériové periferní rozhraní
SSH	-	(Secure Shell) Zabezpečený komunikační protokol
SW	-	(Software) Programové vybavení počítače
Target	-	Cílové zařízení
TCP	-	(Transmission Control Protocol) Komunikační protokol pro přenos dat
TimeStorm	-	Vývojové prostředí např. pro i.MX31
TFTP	-	(Trivial File Transfer Protocol) Protokol pro přenos souborů
UART	-	Asynchronní sériové rozhraní
UML	-	(Unified Modeling Language) Nástroj pro softwarové inženýrství
UNIX	-	Typ operačního systému
USB	-	(Universal Serial Bus) Universální sériová sběrnice
Wi-Fi	-	(Wireless Fidelity) Technologický standard bezdrátové komunikace
Widget	-	Základní prvek grafického uživatelského rozhraní (např. tlačítko, textové políčko)
Windows	-	Operační systém (případně Windows CE, který je pro embedded)
X11	-	Nástroj pro vytvoření grafického rozhraní (např. v Linuxu)
ZigBee	-	Bezdrátová komunikační technologie

Obsah

1 Úvod

Roboti se pomalu stávají každodenní součástí našeho života. Vykonávají různě složité úkoly a pomáhají tak člověku dosáhnout lepší efektivity práce. Pomáhají v místech, která mohou být nedostupná nebo dokonce životu nebezpečná. Nebo naopak mohou rozšířit obzor v neprobádaných oblastech. Příkladem může být průzkumné robotické zařízení.

Mohou nastat situace, kdy člověk potřebuje vidět více i za sebe. Některá místa mohou být skryta lidskému oku, nebezpečná nebo příliš vzdálená. Tam je potřeba vybavit uživatele nějakými zařízeními, aby zvýšily jeho obzor. Tyto mobilní zařízení umožňují uživateli získat více informací o těchto vzdálených místech. Jedním z mobilních zařízení může být průzkumný robot. Jeho nezbytnou součástí je menší ruční mobilní zařízení, které poskytuje uživateli HMI. Pomocí něj může uživatel na dálku získávat informace o prozkoumané oblasti nebo může vysílat příkazy, které má robot vykonávat.

Tato práce se zabývá návrhem a implementací vzdáleného řídicího systému pro mobilní robotické zařízení, zvané také jako průzkumné vozidlo. Vzdálený systém bude získávat naměřené údaje z průzkumného vozidla, vyhodnocovat je a zobrazovat na ovládacím panelu - displeji. Bude také poskytovat funkce pro ruční ovládání vozidla. Komunikace bude probíhat bezdrátově.

Práce vychází z předchozích zkušeností vývoje ovládacího panelu. Stav současného řešení je popsán v kapitole 2, na kterou navazuje kapitola 3 s cíly diplomové práce.

V kapitole 4 je popsán zjednodušený návrh průzkumného vozidla se všemi jeho moduly. Podrobněji je zde probrána problematika návrhu vzdáleného řídicího systému, včetně blokového schématu.

Pro seznámení s hlavní řídicí jednotkou, vývojovou deskou i.MX31 LiteKit firmy Freescale na platformě ARM, slouží kapitola 5. Tato vývojová sada tvoří jádro řídicího systému, ke kterému jsou připojeny další periferie. Volba těchto periferií, tedy zařízeních pro uživatelský přístup (displej, joystick), a jejich oživení je probrána v kapitole 6.

Grafická aplikace potřebuje pro svůj vývoj a spuštění grafické knihovny. Existuje mnoho grafických knihoven, které je možno použít, avšak ne každá z nich podporuje stejný operační systém, platformu nebo programovací jazyk. Srovnání několika grafických knihoven je v kapitole 7. Podrobněji je rozebrána nejpoužívanější grafická knihovna Qt embedded, která je optimalizována pro vestavěné (embedded) systémy.

Problematikou volby systémových nástrojů se zabývá kapitola 8. Jsou zde popsány možnosti použití operačního systému, programovacího jazyka a k ní volba grafické knihovny. Objasněn je i způsob vykonávání grafických aplikací. V závěru kapitoly jsou ukázány možnosti vývoje grafické aplikace včetně postupu vytvoření nového projektu ve vývojových prostředích.

Vzdálený řídicí systém komunikuje s průzkumným vozidlem prostřednictvím bezdrátové sítě WiFi. Srovnání ostatních bezdrátových způsobů připojení je v kapitole 9. Je zde i návod na vytvoření komunikačního spojení přes WiFi moduly OWSPA311 včetně jejich parametrů.

V nejrozsáhlejší kapitole 10 je popsána hlavní práce, tedy návrh a realizace aplikace s graficko-uživatelským rozhraním. Aplikace poskytuje přehledné zobrazení údajů na dotykové obrazovce. Každá z obrazovek ovládacího panelu, využívá pro zobrazování dat různé grafické prvky. Všechny tyto grafické prvky, včetně popisu jejich funkce, jsou podrobněji rozebrány v této kapitole. Pro snazší pochopení jsou k dispozici obrázky. Závěr kapitoly je věnován optimalizaci grafických operací a struktuře programu hlavní aplikace.

Vzdálený řídicí systém implementuje několik řídicích procesů. Této problematice se věnuje kapitola 11. Některé z těchto procesů se nacházejí i v řídicí jednotce průzkumného vozidla. Je zde nastíněn i způsob vykonávání programu ovládacího panelu spolu s vývojovým diagramem. Poslední část kapitoly je věnována režimům řízení průzkumného vozidla a struktuře hlavního programu.

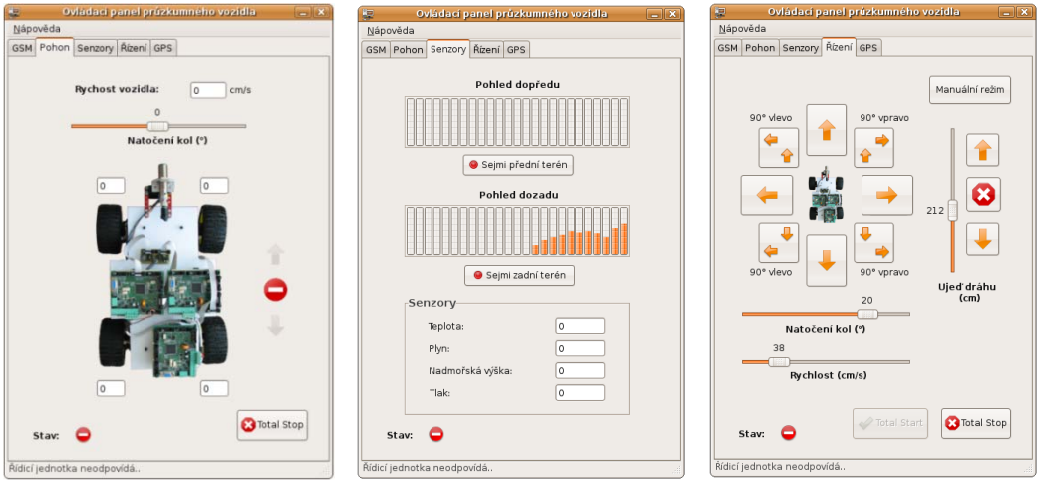
Poslední kapitola 12 se zabývá návrhem krabičky pro uložení vzdáleného řídicího systému. Na závěr už jsou pouze shrnuty dosažené cíle a výsledky.

2 Současný stav

Diplomová práce vychází z předchozích zkušeností bakalářské práce, která se zabývala návrhem a implementací ovládacího panelu pro průzkumné vozidlo. Panel sloužil pro vyhodnocení a zobrazení naměřených údajů a pro manuální řízení vozidla. Vozidlo je na obrázku níže (Obr. 1). Bylo vyrobeno z RC modelu a je složeno z několika modulů s mikrokontroléry HCS12. Pohání jej elektrický pohon a připojené senzory poskytují potřebné informace.

Obr. 1 Pohled na průzkumné vozidlo

Aplikace pro řídicí systém průzkumného vozidla byla vytvořena v operačním systému Linux s použitím programovacího jazyka C a běžela na stolním počítači. Komunikovala s vozidlem prostřednictvím GSM sítě. GSM modul byl připojen k sériovému rozhraní počítače. Náhled aplikace je ukázán na třech obrázcích pod textem (Obr. 2).



Obr. 2 Současný stav ovládacího panelu

S návrhem ovládacího panelu se však vyskytly problémy, které omezovaly jeho použití a funkce.

Seznam hardwarových a softwarových nedostatků:

- Nepříliš šetrné manuální řízení pomocí myši a klávesnice.
- Aplikace běží pouze jako jedno vlákno a v případě chyby celá selže.
- Neexistuje zpětná vazba od vozidla, zda dojelo na místo určení.
- Stolní počítač příliš přesahuje nároky samotné aplikace a není potřeba tak velkého výpočetního a grafického výkonu.
- PC není přenosný (mobilní), vyžaduje 230V zdroj.
- Bezdrátová komunikace vyžaduje pro svůj provoz aktivní SIM kartu.
- Při výpadku GSM spojení je nutné vždy provést startovací sekvenci pro opětovné navázání komunikace.

3 Cíle

Diplomová práce navazuje na předchozí práci ovládacího panelu pro průzkumné vozidlo. Snaží se osamostatnit řídicí systém od standardního stolního provedení, který omezuje obsluhujícího operátora v rozhodování a lepšího pochopení zadané situace. Bude-li systém malý a vestavěný, snižují se i požadavky na výpočetní výkon a spotřebu celého systému.

Vozidlo, které bylo sestaveno z modulů řízení, senzoru a pohonu, je výsledkem týmové práce. Každý modul zde plní důležitou a nepostradatelnou funkci, jako samotná vzdálená vizualizace. Není však vždy podmínkou přítomnost vzdáleného řídicího systému, protože vozidlo je částečně nezávislé při plnění automatického úkolu.

Celá architektura je pak chápána jako distribuovaný řídicí systém, kde na samém vrcholu se nachází vzdálené řízení. Pod ním je řídicí jednotka průzkumného vozidla, která odesílá své požadavky ostatním modulům. Tyto moduly zase získávají cenné informace z připojených senzorů.

Cílem diplomové práce je tedy vytvořit vzdálený řídicí systém na platformě, která bude nejen mobilní, ale také dostatečně silná na to, aby umožňovala navázání komunikace s průzkumným vozidlem, sběr a vizualizaci naměřených údajů a navíc poskytovala operátorovi funkce pro pohodlné řízení.

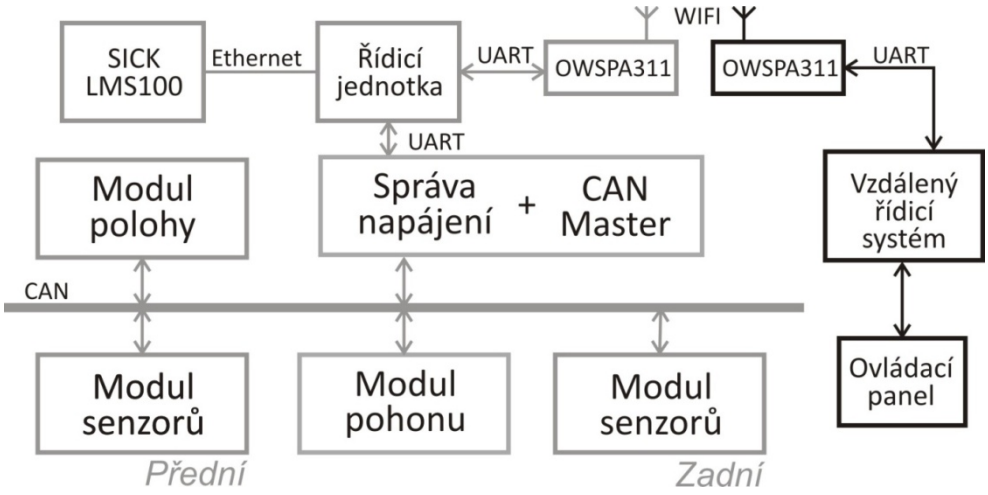
Je nutné vybrat vhodný hardware, který v sobě zahrnuje platformu pro řídicí systém, periferii pro zajištění bezdrátové komunikace, displej a zařízení pro manuální řízení. Dále vybrat operační systém a do něj implementovat vhodnou grafickou knihovnu. A na závěr vytvořit grafickou aplikaci a další řídicí algoritmy, které budou zajišťovat komunikaci, vizualizaci a řízení.

4 Návrh řešení

V kapitole bude stručně popsána problematika průzkumného vozidla a podrobněji problematika návrhu vzdáleného řídicího systému. Znalost průzkumného vozidla je důležitá pro vytvoření seznamu požadavků a samotného návrhu ovládacího panelu.

4.1 Zjednodušené blokové schéma průzkumného vozidla

Řízení průzkumného vozidla je koncipováno jako distribuovaný řídicí systém. Ten je složen z několika samostatných modulů, které jsou umístěny přímo na vozidle a většina z nich spolu komunikují prostřednictvím sběrnice CAN. Řídicí jednotka, tedy hlavní modul vozidla, není připojena ke sběrnici CAN, ale pomocí sériového rozhraní. Vzdálený řídicí modul tvoří systém s vizualizací, který komunikuje s robotickým zařízením (vozidlem) prostřednictvím sítě WiFi a slouží pro sledování vozidla a jeho manuálního ovládání. Na obrázku níže (Obr. 3) je zobrazen blokový návrh řešení dle jejich funkce jednotlivých modulů.



Obr. 3 Blokové schéma robotického zařízení a vzdáleného řízení

Řídicí jednotka - Úkolem řídicí jednotky je navázání spojení se vzdáleným ovládacím panelem, výběr a výpočet nejvhodnější trasy k zadanému cíli, vyhodnocení překážek před a za vozidlem. Prostřednictvím sběrnice ethernet je k jednotce připojen laserový senzor SICK LMS100 pro měření vzdálenosti překážek kolem vozidla. Řízení vozidla se bude provádět prostřednictvím zadávání příkazů ostatním (podřízeným) modulům.

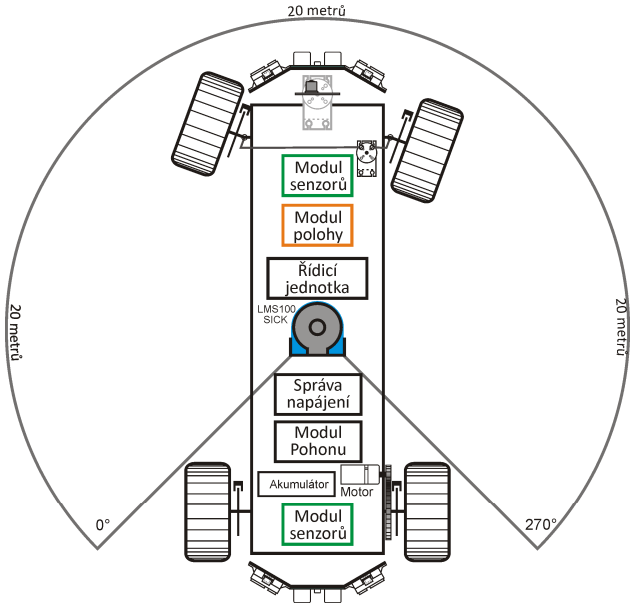
Správa napájení – Slouží pro distribuci elektrické energie z akumulátoru do jednotlivých modulů na vozidle. Řídí nabíjení, dobíjení a sleduje stav baterie při vybíjení a dále jej vyhodnocuje. Je také datovým mostem mezi CAN sběrnicí a nadřazeným systémem.

Modul polohy - Poskytuje řídící jednotce informace o aktuální pozici vozidla. Tu vypočítává z globálních a relativních informací o poloze z připojených senzorů. Mimo to získává informace o okolním prostředí jako teplota, tlak, koncentrace výbušných látek**.**

Modul senzorů - Na vozidle se nacházejí dva tyto moduly. Ty na vozidle fungují jako sběrné uzly, které shromažďují informace ze svého okolí a poskytují je ostatním modulům. Modul nacházející se na přední části vozidla zároveň ovládá natočení předních kol.

Modul pohonu – Obsahuje výkonovou část pro řízení pohonu. Implementuje algoritmus pro regulaci otáček motoru, proudové omezení atd. Požadovaná rychlost pojezdu je ovládána prostřednictvím příkazů z nadřazeného systému.

Vzdálený řídicí systém – Vizualizace a správa vozidla. Data z řídicí jednotky vozidla jsou přenášena prostřednictvím sítě WiFi a použita pro kontrolu, zadávání nových parametrů a jejich vizualizaci na ovládacím panelu (dotykovém LCD displeji).



Obr. 4 Rozmístění komponent na vozidle

Na obrázku Obr. 4 je zobrazeno navržené rozmístění komponent na průzkumném vozidle. Moduly jsou umístěny tak, aby se omezila délka signálových vedení k jejich senzorům.

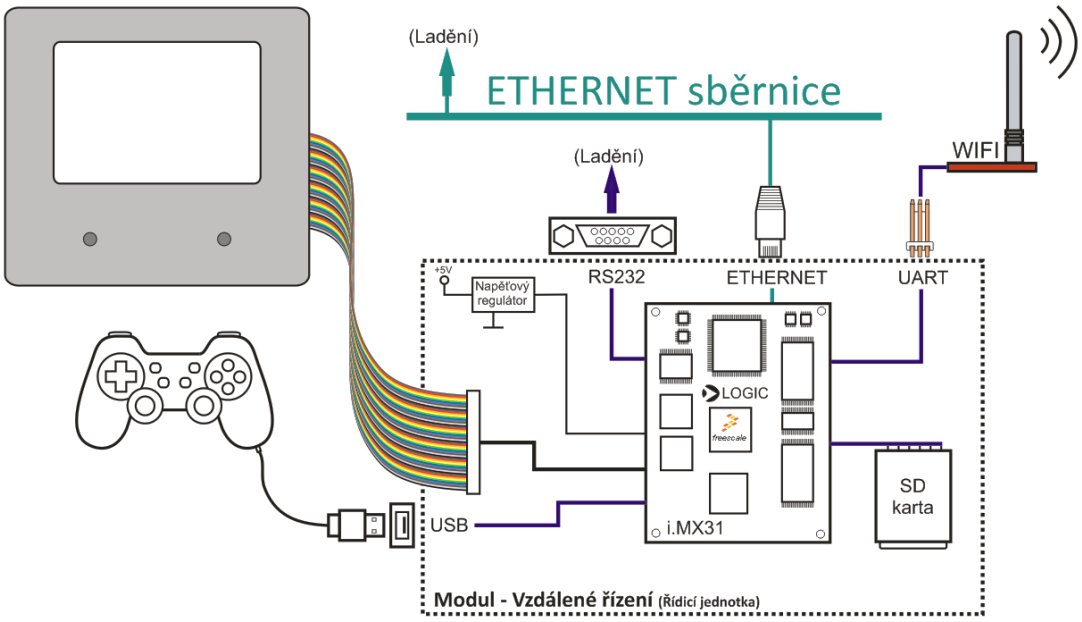
4.2 Návrh vzdáleného řídicího systému

Tato kapitola se zabývá návrhem řešení celého vzdáleného řídicího systému. Prvním krokem je vytvoření seznamu požadavků. K tomuto účelu byl vytvořen UML model. Jelikož je schéma velice rozsáhlé, bylo přidáno do příloh (Příloha IV). Jsou-li známy požadavky, je nutné:

- Zvolit vhodný hardware
- Oživení hardwaru
- Softwarová implementace (operační systém)
- Výběr grafické knihovny a její implementace
- Vytvoření grafické aplikace společně s ostatními procesy
- Testování
- Zapouzdření HW

Hardwarové řešení je zobrazeno na Obr. 5. Volba hardwaru v sobě zahrnuje řídicí jednotku, dotykový LCD displej, joystick (gamepad) a WiFi modul. Komponenty jsou vybrány s ohledem na přenositelnost a snadnou obsluhu operátora (proto dotykový displej a joystick).

Řídicí jednotkou je vývojová deska i.MX31 s výkonným procesorem Freescale, ke kterému jsou připojeny ostatní periferie [2]. Např. k USB rozhraní je připojen joystick (gameapad) a k paralelnímu rozhraní LCD displej. WiFi modul, zajišťující komunikaci s průzkumným vozidlem, je připojen k rozhraní UART. Samotná vývojová sada v sobě neintegruje WiFi rozhraní. Ethernet sběrnice bude sloužit pro nahrávání aplikace do paměti a vzdáleného ladění, podobně jako RS232. Použité komponenty jsou podrobněji popsány v následujících kapitolách.



Obr. 5 Blokové schéma vzdáleného řídicího systému

SD paměťová karta bude plnit funkci úložného prostoru, na kterém se bude nacházet souborový systém s grafickou aplikací.

Hardware je nutné oživit a s tím souvisí i volba vhodného operačního systému, jedná-li se o řídicí jednotku. Operační systém z toho důvodu, že zvyšuje spolehlivost a množství poskytnutých funkcí. Jakmile je tato úloha dokončena, vybere se vhodná grafická knihovna. Zde jsou kladeny požadavky na jednoduchost a nepříliš velké nároky na procesní výkon. Zabere-li knihovna všechen výpočetní výkon, nebude to příznivé pro ostatní běžící procesy.

Následuje tvorba samotné grafické aplikace, ostatních procesů a na závěr testování. Více procesů obecně zvyšuje spolehlivost systému v okamžiku, kdy jeden z nich selže. Posledním krokem je zapouzdření všech komponent do krabičky.

Všechny tyto kroky budou podrobněji popsány a rozebrány v následujících kapitolách.

5 Platforma ARM Freescale i.MX31 LiteKit

Jak již bylo zmíněno, řídicí jednotkou vzdáleného řídicího systému je vývojová deska firmy Freescale i.MX31 LiteKit. Jedná se o odlehčenou verzi na rozdíl od plnohodnotných vývojových desek ADS nebo PDK. Tomu však neubírá výkonu. Vývojová deska se skládá z několika desek. Základním jádrem je deska SOM-LV s výkonným procesorem Freescale, pamětmi a dalšími důležitými obvody. Tato deska je umístěna na desku BaseBoard, která rozšiřuje procesor o periferie, jako jsou USB konektory, ATA konektor pro HDD, konektor pro displej, ethernetový port, konektor RS232 (slouží pro ladění), napájecí konektor, audio vstupy výstupy, slot pro CompactFlash nebo SD paměťovou kartu. Připojením další rozšiřující desky lze vyvést ještě další periferie, jako jsou USB porty, SPI nebo sériové. [3]

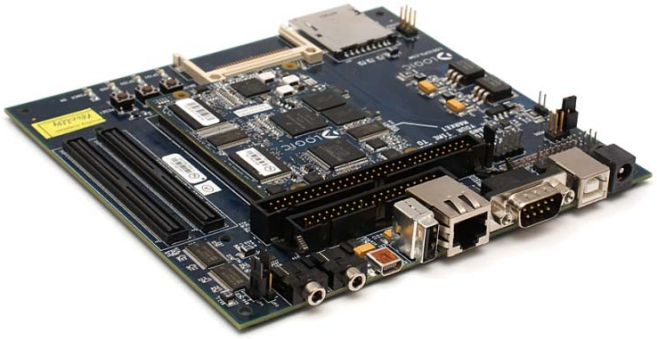
Vývojová deska tedy plně uspokojí potřeby programátora. Lze do něj nahrát operační systém a tím dosáhnout vysokého výkonu.

5.1 Základní parametry

Jako každá vývojová sada, tak i.MX31 lze částečně srovnat s jinými sadami nebo dokonce s klasickým počítačem podle jeho výkonnostních parametrů. Mluví se především o parametrech výkonu procesoru, velikosti a rychlosti pamětí (např. RAM nebo FLASH), šířky sběrnice, či množství podporovaných periferií. [2]

Parametry i.MX31 LiteKit:

- Frekvence procesoru 532 MHz
- Paměť SDRAM 64 MB
- Paměť NAND 64 MB
- Paměť NOR 2 MB



Obr. 6 Vývojový kit i.MX31 LiteKit

5.2 Oživení vývojové desky i.MX31 s podporou grafiky

Diplomová práce v sobě zahrnovala proces oživení desky. Oživení vývojové desky i.MX31 je poměrně složitý a náročný proces vyžadující znalosti Linuxu a vestavěných řídicích systémů. Proč byl vybrán zrovna operační systém Linux, je popsáno v kapitole Operační systém. Kompletní návod pro oživení je rozsahově velmi obsáhlý, a proto je přiložen v příloze (Příloha V). S pomocí tohoto dokumentu může i ne příliš odborný programátor připravit svou vývojovou desku k použití. Zde budou uvedeny pouze rámcové a základní informace o tomto procesu. Pro oživení desky je třeba mít připraveno následující:

- Zavaděč, který je vypálen do i.MX31
- Zkompilované jádro operačního systému Linux
- Zkompilované grafické knihovny pro danou architekturu (armv6j)
- Zkompilované knihovny pro práci s dotykovým displejem (tslib)
- Souborový systém (RFS)
- Vlastní program

Toto všechno je třeba mít připraveno pro cílové zařízení (target) i.MX31. Dále je třeba mít připraven vývojový počítač, který musí mít nainstalovány tyto aplikace.

- Vývojové prostředí (pro cílové zařízení)
- Vývojové prostředí s podporou grafické knihovny Qt (je možná integrace v již stávajícím vývojovém prostředí)
- Křížový kompilátor (Cross-compiler) pro cílové zařízení
- TFTP server pro předávání dat cílovému zařízení
- NFS server pro připojení souborového systému přímo z vývojového PC
- Terminál, SSH pro připojení k cílovému zařízení

Celý systém funguje tak, že po zapnutí i.MX31 je spuštěn zavaděč, který je nakonfigurován, aby si z hostitelského (vývojového) PC stáhnul jádro (kernel) operačního systému a spustil jej. Jádro má v sobě uloženo, kde má hledat souborový systém. Ten se nachází opět na hostitelském PC a z něj si operační systém souborový systém připojí.

Do souborového systému se nahrají zkompilované knihovny pro grafiku a dotykovou obrazovku. Tyto knihovny jsou k dispozici i na hostitelském PC pro vývojové prostředí a jsou nutné pro spuštění grafické aplikace.

Co se týče vývoje vlastní aplikace a jejího ladění, to je prováděno ve vývojovém prostředí. Pokud jsou použita dvě vývojová prostředí, tak jedno pro vývoj standardní aplikace a druhé pro vývoj grafické aplikace. Jelikož se fyzicky souborový systém nachází na hostitelském PC, stačí aplikaci zkompilovat přímo do souborového systému, který má i.MX31 připojen. Poté se stačí k desce připojit (např. pomocí terminálu přes RS232 či SSH) a aplikaci spustit popř. odladit.

Je-li celá aplikace (všechny aplikace) hotová a není potřeba dalších úprav, nahraje se jádro operačního systému do paměti NAND. Aby se jádro samočinně spustilo po startu a naběhnul tak operační systém, přidá se do zavaděče (v paměti NOR) spouštěcí skript. Posledním krokem je zkopírování souborového systému (RFS) s aplikací do paměti i.MX31. [1]

Z důvodu ladění a vytváření aplikace pro i.MX31 bylo výhodné, aby se RFS (root file system) nacházel přímo na hostitelském PC. Odpadala tím potřeba neustálého kopírování dat z PC do i.MX31 a zpět. Problém nastal ve chvíli, kdy byla deska odpojena od PC, jelikož nedisponoval žádnou velkou statickou pamětí, do které by se dala všechna data RFS uložit.

Řešení je v použití externích paměťových médií jako je CompactFlash, HDD nebo SD karet. Pevný disk (HDD) je nákladným řešením a není nutná tak velká kapacita dat. CompactFlash využívá stejně jako HDD paralelní zápis dat, avšak i.MX31 podporuje pouze pomalejší CF a s nepříliš velkou kapacitou. SD karta využívá sériový zápis dat, je levná a snadně dostupná. Z těchto důvodů byla vybrána SD karta pro uložení systému souborů. Informace o RFS jsou také uloženy ve spouštěcím skriptu. Tím je deska osamostatněna od hostitelského počítače. Návod jak zkopírovat systém souborů na SD kartu a vytvoření spouštěcího skriptu je taktéž popsán v dokumentu v příloze (Příloha V).

6 Zařízení pro uživatelský přístup

Zařízení pro uživatelský přístup jsou důležitá pro programátora či operátora, aby mohl přistupovat ke své aplikaci. U osobního počítače je tomu klávesnice, myš, případně tiskárny, joysticky nebo jiná zařízení.

U embedded (vestavěných) zařízení jsou tyto možnosti omezené, jelikož se nejedná o klasický stolní počítač ale tzv. zjednodušenou verzi s několika periferiemi. Proto je nutné se alespoň z části věnovat výběru těchto zařízení, protože ne všechna jsou podporována.

Součástí diplomové práce bylo vybrat tyto periferie a oživit je.

6.1 Dotykový displej

Volba dotykového displeje usnadňuje operátorovi obsluhující panel práci v nutnosti používání klávesnice a myši. Rozmístění tlačítek či jiných prvků je zcela na programátorovi. Výhodou dotykového displeje je možnost připojení dalších jiných periferií, zatímco klávesnice, myš a běžný displej by obsadily tři sloty na vývojové desce.

I.MX31 LiteKit neklade velké požadavky na připojený displej, protože je schopen mu dodávat jak napájení, tak samotné řízení, protože v sobě obsahuje ovladače a paměť (framebuffer). Framebuffer je paměť, do které se uloží data, která jsou následně zobrazena na displeji. Obvykle jsou tyto paměti dvě, kdy do jedné jsou ukládány data a do druhé jsou ve stejném okamžiku data posílána na displej.

Existuje několik způsobů komunikací s displejem. Data o jednotlivých barvách mohou být posílána sériově nebo paralelně, případně ještě jiným způsobem. Vývojová sada umožňuje pouze připojení displeje s paralelním zpracováním barev.

Co se týče podsvícení, existují dvě základní provedení LED a katodové trubice. LED diodové podsvícení se používá hlavně u displejů menších rozměrů, pro větší rozměry je obvyklejší použití CCFL katodových trubic. Jejich nevýhodou je nutnost zdroje vysokého napětí (invertorů).

S ohledem na možnosti i.MX31 byl vybrán 6.4“ VGA Display Kit. Jedná se o displej firmy Logic, který má plnou podporu vývojové desky. Součástí displeje je čtyř-vodičová odporová dotyková fólie. Odporová fólie sice není moc komfortní řešení, ale pro aplikaci vystačuje. [2]

6.1.1 Základní vlastnosti

Displej se připojuje k vývojové desce i.MX31 prostřednictvím 60 pinového konektoru. Tento konektor poskytuje displeji také napájení (5V) a není nutný přídavný zdroj. Podsvícení displeje je realizováno katodovými trubicemi, které jsou napájeny zdrojem vysokého napětí invertoru. LCD displej, napájecí obvod inverotru a dotyková fólie jsou uloženy v hliníkovém obalu, který jej také

chrání. Mezi dotykovou fólií a LCD displejem je tenká skleněná vrstva, které brání proti poškrábání.

Základní parametry:

- Úhlopříčka 6.4 palce
- Rozlišení 640x480 bodů (pixelů)
- 262 tisíc barev
- Odporová dotyková fólie

Obr. 7 Dotykový displej Logic

6.1.2 Oživení displeje

Budou zde uvedeny kroky, které bylo nutné provést pro oživení displeje.

- Připojení k i.MX31 LiteKit 60 pinovým konektorem
- Kompilace jádra s podporou grafického displeje, framebufferu a dotykovou obrazovkou
- Pro správnou kompatibilitu dotykové obrazovky nutná verze jádra 2.6.24
- Softwarové zapnutí displeje
- Otevření dotykové obrazovky v Linuxu pro kalibraci
- Případné přiřazení události pro dotykovou obrazovku

Pokud je přidána podpora displeje a dotykové fólie v jádře operačního systému, měl by se displej objevit v zařízeních `/dev` jako `fb` (framebuffer) a fólie v `/dev/input` jako `event0`. Ve starší verzi jádra (2.6.22) se objevil problém s dotykovou obrazovkou, kdy určitá plocha na fólii nebyla aktivní. Tento problém vyřešila novější verze jádra.

Před spuštěním operačního systému je nutné displej softwarově zapnout příkazem:

```
video-open 5 16
```

Pokud je vše v pořádku, mělo by se po spuštění systému zobrazit v levém horním rohu obrazovky logo operačního systému. V opačném případě bude displej stále vypnutý.

6.1.3 Oživení dotykové obrazovky

Dotyková obrazovka je sice připojená k vývojové desce a v operačním systému, avšak pro její použití jsou nutné určité knihovny. Jinak se chová pouze jako obyčejné zařízení. Do operačního systému se musí nakopírovat zkompilované knihovny (přeložené pro danou architekturu, v našem případě ARM) pro dotykovou obrazovku. Pak je možné dotykovou fólii kalibrovat nebo ji používat pro manipulaci s grafickými aplikacemi.

Kompilace těchto knihoven a návod jak je implementovat do systému je opět podrobněji uveden v dokumentu v příloze (Příloha V).

6.2 Joystick (Gamepad)

Dotykový displej usnadňuje uživateli (operátorovi) práci při obsluze grafické aplikace. Při manuálním řízení vozidla jsou však jeho funkce nedostačující a proto je potřeba vybrat jiné řešení. Nabízí se joystick, který svým snadným ovládáním přímo vybízí k použití řízení vozidla. Kromě joysticku lze použít i gamepad, ten má výhodu dvou analogových ovladačů.

Joystick neklade velké požadavky, co se týče na připojení k vývojové desce, protože se chová jako standardní zařízení a připojuje se přes USB rozhraní.

Obr. 8 Joystick Predator

Základní parametry:

- Připojení přes USB
- Dvouosý analogový ovladač
- Analogové kolečko
- 4 tlačítka
- jedno čtyřmístné tlačítko

6.2.1 Oživení joysticku (gamepadu)

Opět zde budou uvedeny pouze kroky, které bylo nutné provést pro oživení joysticku.

- Připojení k i.MX31 LiteKit USB portu
- Kompilace jádra s podporou HID zařízení, joysticku a USB (gamepadu)

Je-li správně přidána podpora joysticku v jádře operačního systému, měl by se objevit v zařízeních `/dev` jako `js0`. K joysticku na rozdíl od dotykové folie existuje už vytvořené API v jazyce C. To je zdokumentované a přístupné k nahlédnutí ve zdrojovém kódu operačního systému. K zařízení `js0` se přistupuje jako k souboru s definovanou strukturou, která je uvedena pod textem. Z této struktury lze pak vyčíst, které tlačítko bylo stisknuto, nebo do jaké pozice byl posunut analogový ovladač.

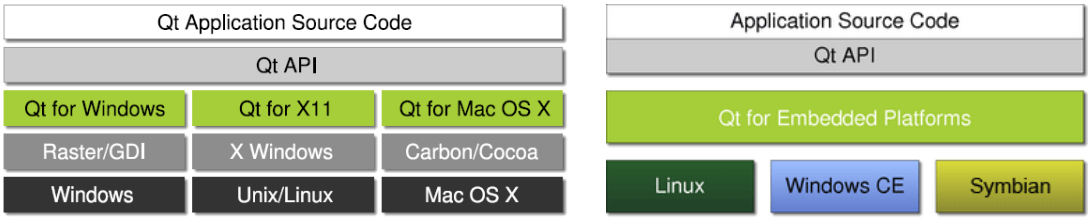
```
struct js_event {
        __u32 time;     // čas události v ms
        __s16 value;    // hodnota
        __u8 type;      // typ události
        __u8 number;    // číslo osy/tlačítka
};
```

7 Grafické knihovny

Každá grafická aplikace potřebuje pro svůj chod grafické knihovny. Možnosti použití těchto knihoven jsou omezené v závislosti na použité platformě. Ne každá knihovna poskytuje programátorovi velké množství funkcí, grafických prvků a plnohodnotnou dokumentaci.

7.1 Qt

Qt je multiplatformní knihovna pro vytváření programů s grafickým uživatelským rozhraním (GUI). Využívá nízko úrovňové API různých platforem, které podporuje (Microsoft Windows, X11, Mac OS X a Embedded Linux). Aplikace mohou být kompilovány a spuštěny pro každou cílovou platformu. [5]

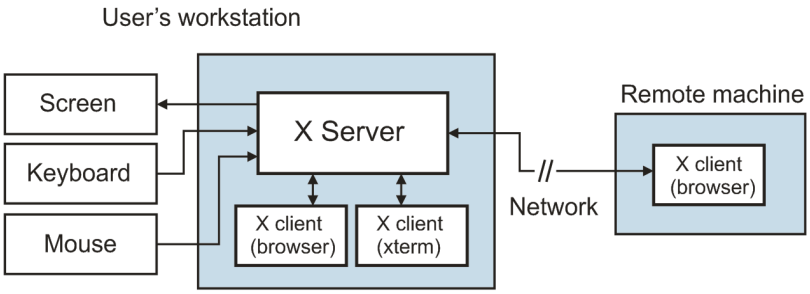


Obr. 9 Modely Qt aplikací pro různé platformy

Qt je dostupná stejně dobře jak pro vestavěné (embedded) platformy, tak pro stolní platformy. Vlastnosti poskytované každou platformou závisí na schopnostech a architektuře operačního systému. Běžně jsou dostupné tři platformy (Linux, Windows CE a Symbian).

7.2 X11

X Window System (X11 nebo X), je softwarový systém a síťový protokol, který poskytuje GUI pro síťové počítače. Implementuje X displej protokol a poskytuje práci s okny v rastrové grafice pro displeje, obsluhuje klávesnici a ukazatele (myš). [6]



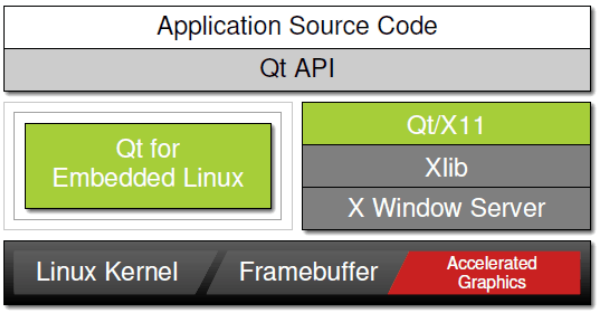
Obr. 10 Blokové schéma X Serveru a aplikace (X klient)

X11 poskytuje základní framework a prvky pro stavbu GUI prostředí: kreslení, posun oken na obrazovce a spolupráce s myší nebo klávesnicí. X11 je postaven jako dodatečná aplikační vrstva na vrchu operačního systému. Byl speciálně navržen, aby byl používán přes síť než jako připojený k zobrazovacímu zařízení. X11 má vlastnosti síťové průhlednosti – zařízení, kde běží aplikace, se může lišit od uživatelského zařízení (server s displejem).

X11 používá model klient-server. Server komunikuje s různými klientskými programy. Přijímá žádosti pro grafický výstup (okna) a odesílá zpět uživatelský vstup (z klávesnice, myši nebo dotykové obrazovky). Klient řídí uživatelské rozhraní.

7.3 Qt embedded linux

Qt embedded linux je C++ framework pro GUI a aplikace vyvinuté speciálně pro vestavěná zařízení. Běží na různých procesorech, obvykle embedded Linux. Poskytuje stejné API a nástroje jako X11, Microsoft Windows, Mac OS X a také zahrnuje třídy a nástroje pro speciální embedded podporu. Má menší systémové požadavky než embedded řešení na X Window systémech (menší úložný prostor Flash a paměť RAM). Obsahuje také negrafické komponenty pro speciální třídy – práce se sítí, podpora více vláken a databáze. [5]



Obr. 11 Model Qt aplikace s Qt embedded Linux

Qt pro embedded Linux šetří paměť, protože nepotřebuje X11 server nebo Xlib, jelikož zapisuje přímo do framebufferu. Zahrnuje framework pro vytvoření jak aplikací, tak prostředí. Výsledkem je, že obsahuje rysy pro okenní manažer, meziprocesní komunikaci a prostředky pro ovládání vstupních a zobrazovacích zařízení.

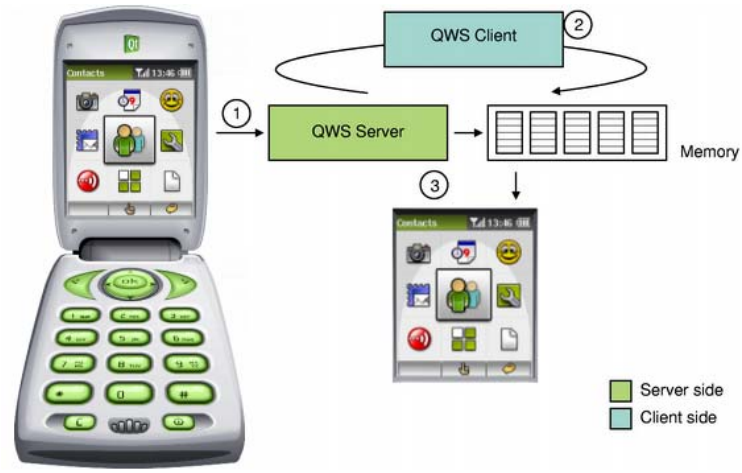
Linux framebuffer je obecně aktivní na všech moderních Linux distribucích. Pro vývoj a ladící účely Qt poskytuje virtuální zásobník (buffer).

7.3.1 Architektura

Qt embedded Linux aplikace vyžadují pro svůj provoz spuštění grafického serveru nebo může být serverem přímo aplikace. Každá Qt pro embedded Linux aplikace může pracovat jako server. Pokud je spuštěna více než jedna aplikace, následující aplikace se připojují k existující server aplikaci jako klienti. Server tedy řídí obsluhu ukazatele, znakového vstupu a výstup na

obrazovku. Navíc server nastavuje vzhled kurzoru a spořiče obrazovky. Klient vykonává všechny konkrétní grafické operace aplikace. Server aplikace je reprezentována instancí třídy QWSServer, zatímco klient třídou QWSClient. Klienti komunikují se serverem přes sdílenou paměť a tím je vzájemná komunikace minimální. Pomocí příkazového řádku může být určitý program spuštěný jako server.

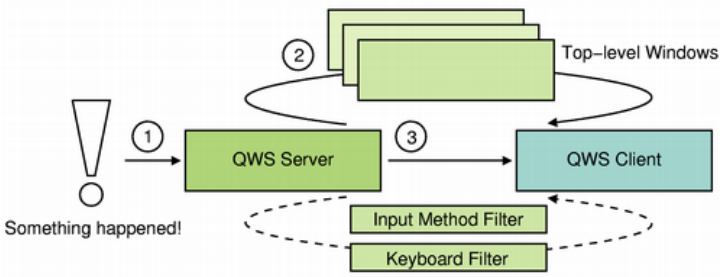
Všechny systémy vytvořené událostmi, obsluhující klávesnici a myš, vstupují do aplikace serveru, která pak odešle události do příslušného klienta. Při generování grafiky, klient ukládá widgety do paměti, zatímco server je odpovědný za vkládání obsahu paměti na obrazovku.



Obr. 12 Model architektury Qt embedded Linux

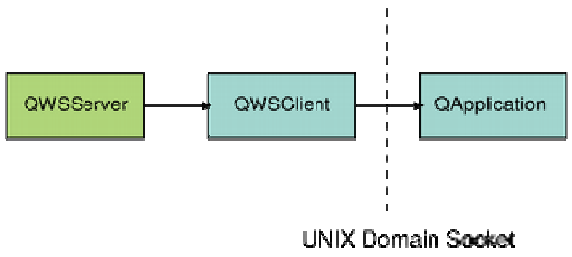
7.3.2 Server – Klient komunikace

Běžící aplikace neustále mění vzhled obrazovky přidáváním a odstraňováním widgetů. Server udržuje informaci o každém úrovni (top-level) okna. Kdykoli server příjme událost, musí přesně zjistit, které okno událost vyvolalo. Každé okno může identifikovat klienta, který ho vytvořil a vrátí jeho identifikační číslo serveru na požádání. Nakonec server předá zapouzdřenou událost klientovi.



Obr. 13 Server – Klient komunikace

Server komunikuje s klientskou aplikací přes UNIX doménu socket. Může znovu získat přímý přístup ke všem událostem, které klient přijal ze serveru. Klienti (a server) komunikují mezi sebou použitím QCOPChannel třídy. QCOP je komunikační protokol pro přenos zpráv na různých kanálech. Kanál je definovaný jménem a tím, kdo chce poslouchat. QCOP protokol povoluje klientům komunikovat se stejným adresovým prostorem a mezi různými procesy.

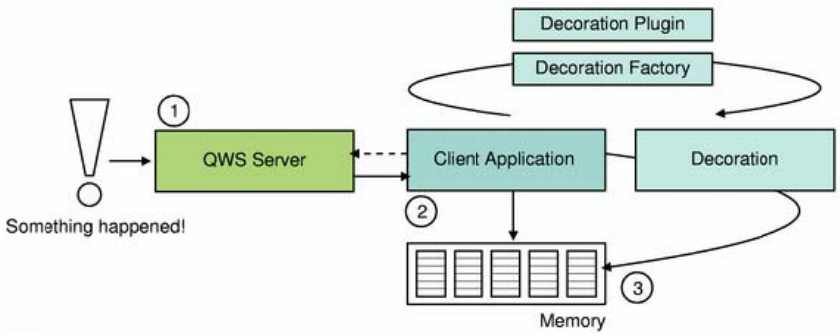


Obr. 14 Komunikace pomocí socketů

Klienti mohou měnit zprávy prostřednictvím QCOP kanálů. Server jednoduše vysílá QCOP zprávy do všech aplikací naslouchající na daném kanálu. Aplikace používá Qt signály a sloty aby odpovídala. Zprávy mohou být doprovázeny binárními daty, obvykle sérií řídicích tříd. Podprocesy Qt poskytují další způsob jak vykonávat asynchronní mezi-procesní komunikaci. Aplikace mohou spouštět externí programy a komunikovat s nimi přes standardní vstupy, výstupy a chybové proudy. Tento způsob může být použit pro asynchronní monitorování podprocesů.

7.3.3 Vytváření grafiky

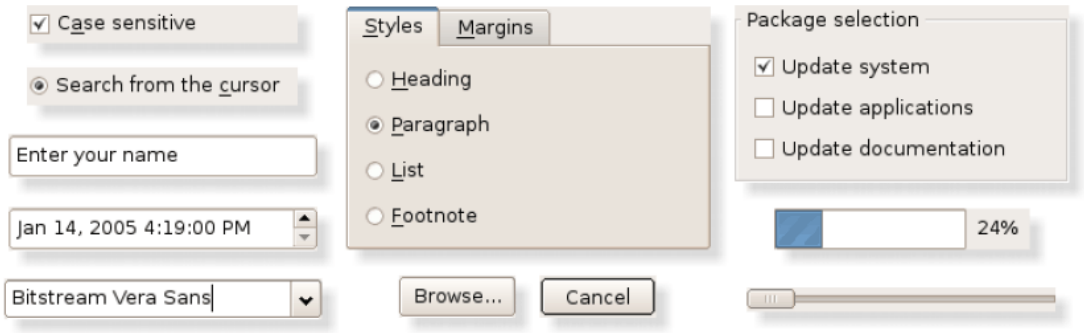
Standardní chování Qt embedded Linux je takové, že každý klient převádí widgety a různé malůvky (decoration) do paměti, zatímco server kopíruje obsah paměti do framebufferu. Kdykoli klient přijme událost, která mění nějaký widget, aplikace aktualizuje důležitou část paměťového zásobníku.



Obr. 15 Model vytvoření grafiky

7.3.4 Graficko uživatelské rozhraní

Qt embedded Linux poskytuje bohatou sadu widgetů pro vytváření aplikací. Widgety jsou vizuální prvky určené pro tvorbu uživatelského rozhraní. Tlačítka, menu, posuvníky, boxy na zprávy a aplikační okna jsou příkladem widgetů. Je možné nastavit automatické rozmísťování, poskytují inteligentní minimální a výchozí velikosti pro okna a automaticky přemísťují widgety, pokud se změní jejich obsah.



Obr. 16 Prvky (Widgety) graficko-uživatelského rozhraní

7.3.5 Vstupní zařízení

Qt embedded linux poskytuje několik protokolů pro myš – BusMouse, IntelliMouse, Microsoft a MouseMan. Qt podporuje standardní 101 klávesovou klávesnici a Vr41XX tlačítka. Dodatečně jsou podporovány i dotykové obrazovky knihovou slib, jiné knihovny od třetí osoby, případně použít knihovnu slib zkompilovanou přímo ve Qt knihovně. Podobně Qt může používat vstupní zařízení podporované od DirectFB, kdy se pro konfiguraci používá DirectFB ovladač obrazovky.

7.3.6 Spuštění aplikace

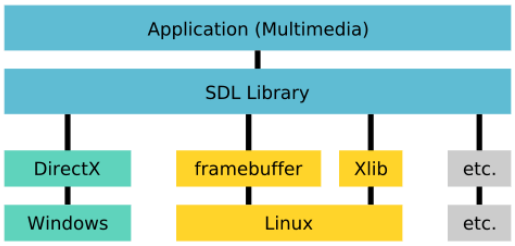
Qt embedded Linux aplikace vyžaduje server aplikaci k tomu, aby mohly být spuštěna, nebo může být samotná aplikace serverem. Každá aplikace může být serverem, pokud je nastaven QApplication objekt s typem QApplication::GuiServer nebo spuštěna aplikace v příkazovém řádku s parametrem `-qws`. Aplikace mohou běžet na jednom nebo více displejích. K tomu jsou dostupné různé nastavení v příkazovém řádku.

7.4 SDL knihovna

Simple DirectMedia Layer (SDL) je křížová platforma (cross-platform) multimediální knihovna navržena k tomu, aby poskytovala nízko úrovňový přístup k audiu, klávesnici, myši, joysticku, 3D hardware OpenGL a 2D video zásobníku. Využívají ji MPEG playback software, emulátory a mnoho dalších populárních her. SDL podporuje Linux, Windows, Windows CE,

BeOS, MacOS, Mac OS X, FreeBSD, NetBSD, OpenBSD, BSD/OS, Solaris, IRIX a QNX. Knihovna je napsaná v C, ale pracuje nativně s C++ a má vazby s několika jinými jazyky jako Ada, Objective C, C#, D, Java, Pascal, Perl, PHP, Python atd. SDL je distribuovaná pod GNU LGPL verze 2. Tato licence dovolí užívat SDL volně v komerčních programech. [8]

SDL má v sobě slovo „Layer“ (vrstva) protože je ve skutečnosti zabalená okolo funkcí operačního systému. Hlavní cíl SDL je poskytnout hlavní framework pro přístup k těmto funkcím. Sousta zdrojových kódů je rozdělena do oddělených modulů pro každý operační systém.



Obr. 17 Model SDL knihovny pro různé platformy

Na Microsoft Windows systému SDL používá DirectX. Pro X11 platformy, zahrnující Linux a OpenVMS používá SDL knihovnu Xlib. aby komunikovala s X11 systémem pro grafiku a události.

7.5 Microwindows

Microwindows je okenní systém, který má plno vlastností, aby byl použit na PC nebo PDA. Nano-X Window Systém je Open Source projekt zaměřený na přinášení rysů moderních grafických prostředí oken pro malé zařízení a platformy. Nano-X dovoluje aplikaci, aby byla sestavena a otestována v linuxu na stolním počítači nebo přes křížový kompilátor na cílovém zařízení. Nano-X Window Systém byl předtím nazýván Microwindows, ale byl přejmenován kvůli konfliktům s Microsoft Windows ochrannou známkou. [7]

Nano-X systém je extrémně přenosný a kompletně napsaný v C, ačkoli některé rutiny byly vytvořené v asembleru pro větší rychlost. Je přenosný do Intelu 16 a 32 bit procesory stejně dobře, jako pro MIPS R4000, StrongARM a PowePC. Nano-X systémy běží na 32 bitových Linux systémech s podporou framebufferem nebo skrz populární SVGAlib knihovnu.

X11 driver dovoluje Microwindows aplikacím aby běžely na klasickém stolním počítači nad X Window systémem. Tento driver emuluje všechny Mircowindows truecolor módy palet, takže aplikace může používat displej cílového zařízení přímo na displeji stolního počítače bez ohledu na vlastnosti displeje stolního počítače.

Microwindows je v podstatě návrh vrstev, který povoluje různým vrstvám, aby byly použity nebo přepsány do sady potřebné pro implementaci. Ovladače obrazovky, myši, touchpadu a klávesnice poskytují nízko-úrovňový přístup k běžným displejům a jiným vstupním zařízením.

Přenosný grafický nástroj (engine) je implementován ve střední úrovni a poskytuje podporu pro kreslení čar, ploch, polygonů a různých barevných modelů. Na nejvyšší úrovni jsou implementována různá API, která poskytují přístup pro grafické aplikace. Tyto API mohou, nebo nemusí poskytovat okenní vzhled. Microwindows je podporován pro Windows Win32/WinCE, GDI a Nano-X API. Tyto API poskytují blízkou kompatibilitu s Win32 a X-Window systémy, dovolující programům se snadno přemístit do jiných systémů.

8 Volba softwarových nástrojů

8.1 Operační systém

Operační systém je kombinací programových funkcí, které dávají programátorům základní nástroje pro návrh a implementaci svých aplikačních programů. Umožňují uživatelům řídit zpracování programů, aniž by potřebovali řídit technické vybavení. Usnadňují uživatelům čtení a zápis dat na paměťová média (disky), tisk výsledků a vůbec celkově ovládat komunikaci mezi aplikačním programem a technickým vybavením. [4]

Rozsáhlé a složité operační systémy mívají mnohdy tolik funkcí a možností, že jich u řady aplikací mnohdy ani nelze ekonomicky využít. Všechny tyto možnosti zvyšují cenu operačního systému. V praxi, zejména v průmyslových aplikacích, vede tento fakt k tomu, že se pro každou aplikaci používá jinak modifikovaný operační systém, aby byl co nejjednodušší a poskytoval potřebné služby.

Použití operačního systému přináší spoustu výhod a usnadnění při vývoji pak vlastní aplikace. U složitějších architektur a výkonnějších procesorů s velkým množstvím periferií by bylo jejich nastavování a obsluha příliš složitá. Jde například o různá komunikační rozhraní (USB, RS232), displeje či paměti. Operační systém může být na míru navržen přímo pro naši architekturu a tím velice usnadní práci při vývoji softwaru.

Volba operačního systému závisí na možnostech jeho použití, bude-li se jednat o systém pro kritické aplikace nebo systém řídicí regulaci teploty v domácnosti. Proto se mohou požadavky na systém značně lišit. V jiných případech jsme omezováni možnostmi samotné architektury, pro kterou má být operační systém použit. Ne na každém zařízení se dá spustit každý operační systém. Příkladem může být nemožnost nasazení operačního systému Windows XP na počítače Apple.

Deska i.MX31 podporuje v zásadě dva operační systémy.

- Timesys Linux
- Windows CE

Existuje zde podpora i operačního systému QNX, ne však přímo pro zjednodušenou verzi desky LiteKit. Pro implementaci by se musel systém upravit.

Z předchozích zkušeností s prací s těmito operačními systémy byl zvolen Linux. Při výběru byly brány v potaz tyto aspekty:

- Spolehlivost OS
- Podpora komunikace mezi procesy
- Možnost jednoduché komunikace pomocí SSH
- NFS Server pro vývoj (šetření počtu přepalů FLASH paměti)
- Velmi silný příkazový řádek

Spolu s vývojovou sadou je dodáván kompilátor (cross compiler) pro C a C++. S tím souvisí další volba, a to je programovací jazyk, ve kterém bude samotná aplikace napsána a vývojové prostředí, ve kterém bude aplikace vytvořena.

8.2 Programovací jazyk

Pod operačním systémem Linux je možné psát ve spoustě programovacích jazyků. Ať se jedná o jednodušší skriptovací jazyky či samotné C, C++ nebo Python. Zde je ukázáno jednoduché srovnání dvou programovacích jazyků, C a C++.

Jazyk C:

- Velmi rychlý a jednoduchý
- Umožňuje programování na velmi nízké úrovní
- I základní typy jako je textový řetězec si uživatel musí napsat sám
- Nepodporuje objekty
- Není přísně typový

Jazyk C++:

- Velmi rychlý a jednoduchý
- Umožňuje programování na velmi nízké úrovni
- Standardní knihovny obsahují základní typy jako je řetězec, list, stream
- Podporuje objekty, třídy
- Je přísně typový
- Rozšířenější u složitějších aplikací než C
- Používanější u grafických knihoven (nejedná-li se o primitivní knihovny)

Než bude zcela rozhodnuto, který programovací jazyk bude použit, je třeba ještě poukázat na možnosti grafických knihoven. Ne každá má podporu obou těchto jazyků.

8.3 Výběr grafické knihovny

Nejdůležitějším úkolem v diplomové práci bylo vybrat vhodnou grafickou knihovnu, protože od ní se bude odvíjet návrh a realizace grafické aplikace ovládacího panelu.

Bylo popsáno několik způsobů jak spustit a vytvořit grafickou aplikaci na embedded zařízení. Je-li aplikace vytvořena grafickou knihovnou Qt, je možné pro její běh použít buď X11 server nebo přímo Qt embedded Linux a tím se práce mnohem usnadní, jelikož nejsou nutné další knihovny X11 a Xlib. Qt embedded pracuje přímo s framebufferem a pak vypisuje obsah paměti na obrazovku. Qt má podporu pro různé platformy jako Windows CE, Linux a Symbian.

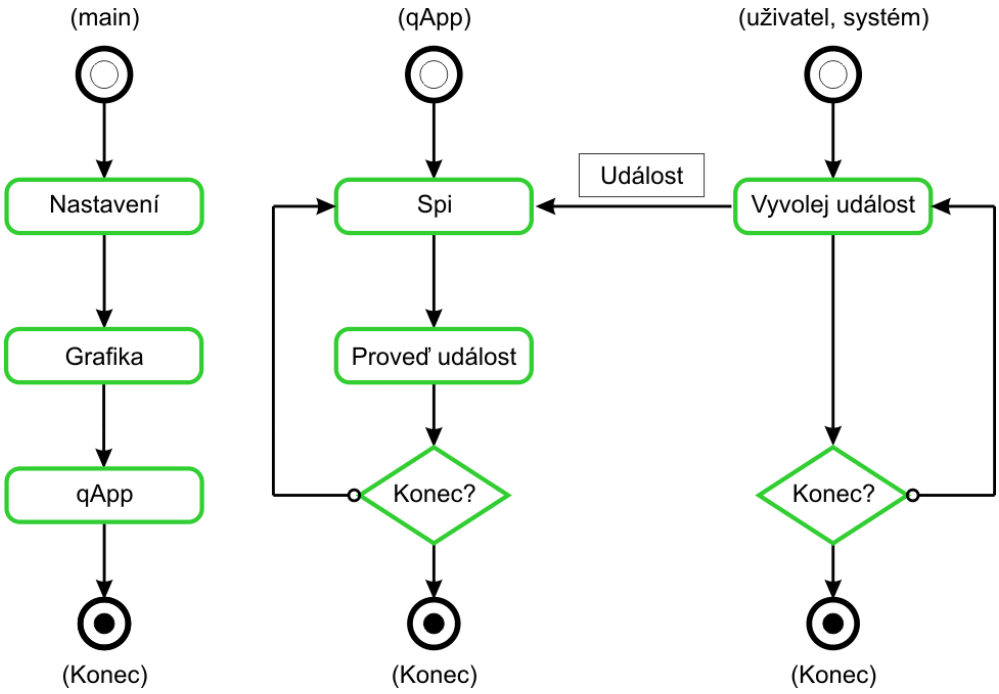
Další možností je použití SDL grafické knihovny. Ta má podobně jako Qt podporu pro různé typy operačních systémů. Pro běh samotné grafické aplikace je možné použít jako u Qt Xlib knihovny a X11 nebo to celé nahradit Nano-X systémem, který je jednodušší a má také velkou podporu pro různé operační systémy.

SDL na rozdíl od Qt je nízko-úrovňová grafická knihovna, takže sestavení aplikace může být zdlouhavější a čím rozsáhlejší by aplikace byla, tím více času by návrh a vývoj zabral. Knihovna je celá napsaná v jazyce C na rozdíl od Qt, která je v C++. Obě knihovny mají podporu i pro jiné programovací jazyky. Qt disponuje návrhářem, ve kterém lze jednoduše vytvořit GUI naskládáním widgetů na plochu. Je to snadnější řešení, než si například zvlášť definovat polohu a rozměry každého tlačítka. Při manipulaci nebo posunutí se musí znova přepsat celý kód. Návrhář tuto práci dělá automaticky a proto je toto řešení výhodnější a rychlejší.

Z důvodu velké podpory, široké dokumentace a snadného návrhu bude použita pro návrh ovládacího panelu grafická knihovna Qt, která používá programovací jazyk C++.

8.3.1 Princip vykonávání grafické aplikace

Obecně je každá aplikace využívající graficko-uživatelské rozhraní proces řízený událostmi. Názorná ukázka takového způsobu řízení je na obrázku pod textem (Obr. 18).



Obr. 18 Popis aktivit hlavního programu

Každý proces začíná základní funkcí **main**, která je nutná v každém programu. Je to první funkce, která je obsloužena hlavním vláknem. Následuje deklarace a definice použitých proměnných. Dalším krokem u grafických aplikací je vytvoření hlavního grafického okna. Podobně jako u funkce **main**, je nutné i zde definovat proměnné (tlačítka, textová pole, obrazovky). Zavoláním funkce **qApp** dojde ke spuštění nekonečné smyčky, která vykonává potřebné grafické operace.

Tato smyčka nezatěžuje neustále procesor, jak by se mohlo zdát. Hlavní vlákno spí do té doby, dokud nenastane určitá událost. Proto se zde mluví o systému řízený událostmi. Událost může být vyvolána uživatelem, například stiskem tlačítka nebo aktivací jiného grafického prvku, a v té chvíli se provede příslušná funkce (v některých grafických knihovnách je tato funkce zvaná jako **callback**, v Qt knihovně zase **slot)**. Nebo může jít o systémovou událost, kdy dojde například k přetečení časovače nebo příchodu dat na sériový port.

8.4 Vývojové prostředí

TimeStorm je vývojové prostředí, které je podporováno přímo pro vývojovou sadu i.MX31. Prostředí je rozšířením vývojového prostředí Eclipse. Lze v něm vytvářet aplikace jak v jazyce C tak v C++. Pro podporu grafických aplikací je nutné přidat Qt plugin (modul), který je běžně dostupný ke stažení z internetu. Qt aplikace vyžadují pro svůj překlad speciální kompilátor. Ten lze získat buď instalací kompletního vývojového nástroje Qt SDK, nebo opět stažením z internetu.

Druhým řešením je použití nástroje Qt SDK, který v sobě obsahuje Qt Creator, Qt Designer a další užitečné nástroje. Qt Designer slouží pouze pro grafický návrh aplikace, rozmístění grafických prvků v okně. Oproti tomu Qt Creator je nástroj k vytváření grafických aplikací s integrovaným návrhářem.

Obě tyto prostředí, Timestorm a Qt Creator, jsou silné nástroje pro vytváření grafických aplikací, avšak Qt Creator má tu výhodu, že obsahuje plnou podporu dokumentace s příkladem použití jednotlivých funkcí a je speciálně navržen pro práci s grafikou. Na druhou stranu TimeStorm je zase vhodný pro standardní aplikace.

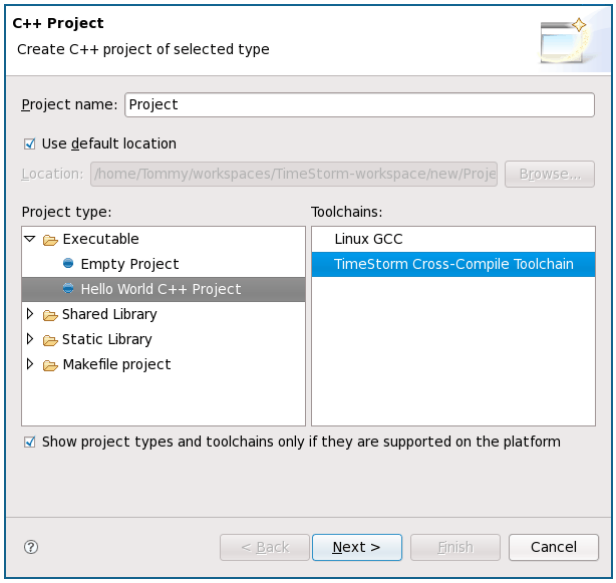
Co se týče autorských práv, tak TimeStorm a Qt SDK jsou vývojová prostředí, na které se nevztahují žádné poplatky v nekomerčním užívání. Licence na TimeStorm se vztahuje pouze v případě, pokud by byla potřeba ladění vývojové desky přes vývojové prostředí.

8.4.1 Vytvoření projektu v TimeStorm

V této kapitole bude popsán postup pro vytvoření nového projektu ve vývojovém prostředí TimeStorm v programovacím jazyce C++.

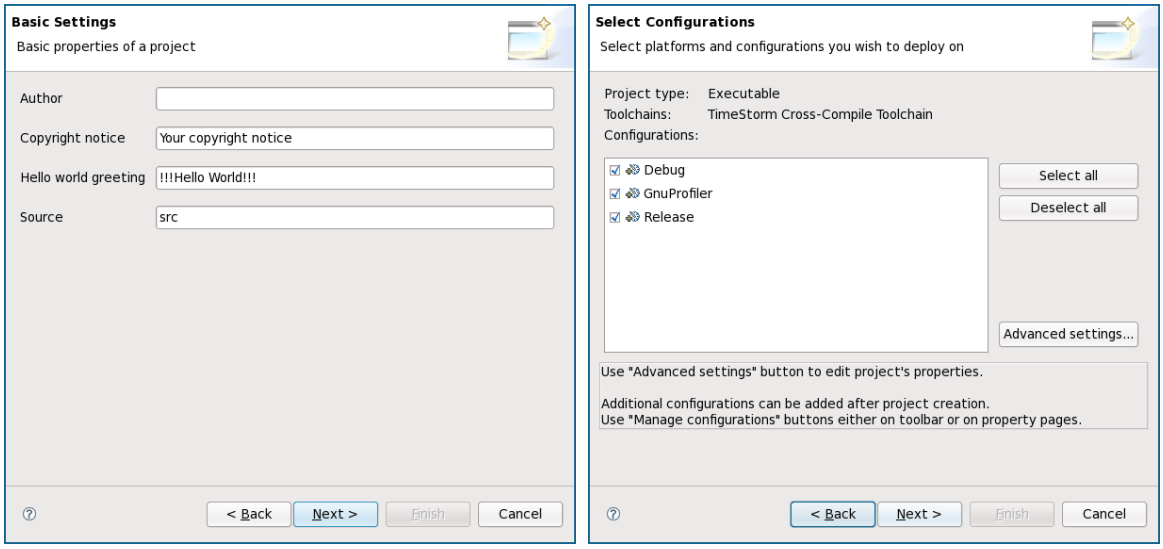
Před vytvořením nového projektu je nutné mít nainstalovaný křížový kompilátor pro danou architekturu, v opačném případě bude možné vytvořit pouze projekt s kompilátorem, který používá operační systém. Návod jak nainstalovat křížový kompilátor je uveden v dokumentu v příloze (Příloha V).

Po spuštění vývojového prostředí se musí vybrat v záložce *File -> New -> C++ Project*. Je zde i možnost importování stávajícího projektu nebo vytvoření projektu Qt. Otevře se nabídka s novým projektem, kde je nutno zadat název projektu, typ a výběr toolchain kompilátoru. Situaci demonstruje obrázek pod textem (Obr. 19).



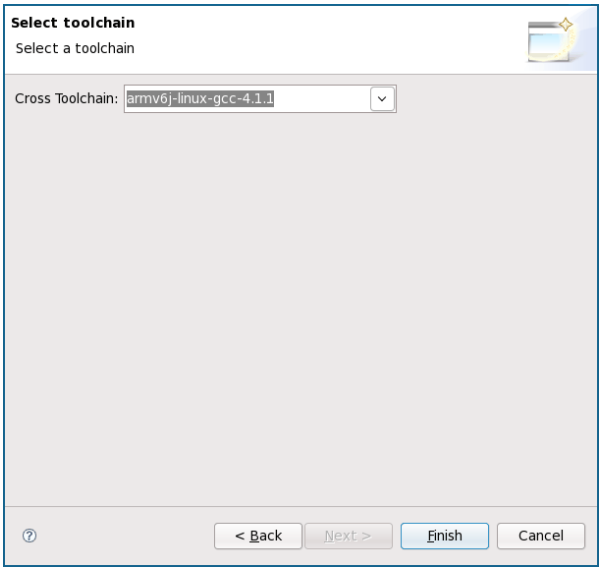
Obr. 19 Nabídka nového projektu v C++

Po kliknutí na *Next* se objeví další nastavení, kde jsou údaje o autorovi. Jsou to pouze informativní údaje a není potřeba je vyplňovat (Obr. 20). Následuje výběr konfigurace. Jako v předchozím případě lze toho nastavení přeskočit.



Obr. 20 Základní nastavení a výběr konfigurace

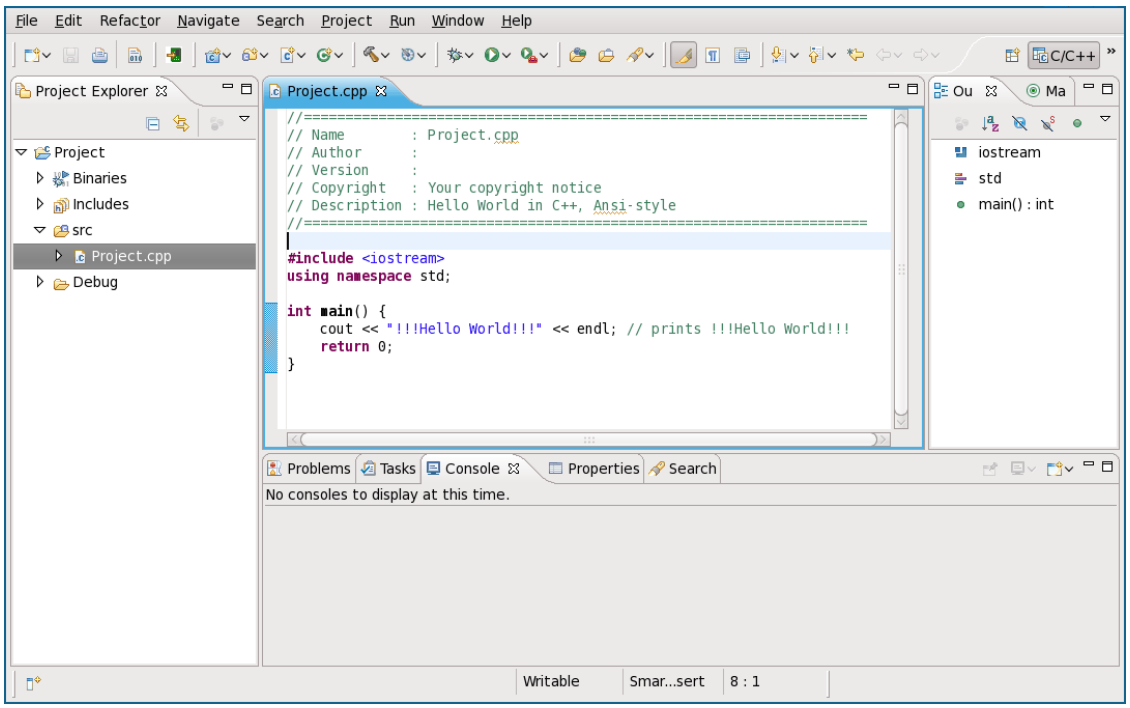
Důležitým parametrem je výběr kompilátoru. Pokud je daný kompilátor v systému správně nainstalován, měl by se objevit v nabídce. S vývojovou sadou je dodáván kompilátor armv6j-linux-gcc-4.1.1. Výběr křížového kompilátoru je na obrázku dole (Obr. 21).



Obr. 21 Výběr křížového kompilátoru

Kniknutím na *Finish* bude vytvořen projekt. Pokud je projekt přeložen s kompilátorem, který je pro jinou architekturu než hostitelský počítač s vývojovým prostředím, je dosti pravděpodobné, že program nebude možné spustit nikde jinde, než na cílové architektuře.

Posledním obrázkem (Obr. 22) je pohled na vývojové prostředí s vytvořeným novým projektem.



Obr. 22 Vývojové prostředí s vytvořeným projektem

8.4.2 Vytvoření projektu ve Qt Creator

Podobně jako v předcházející kapitole bude zde ukázka založení nového projektu, ale s tím rozdílem, že se bude jednat o vytvoření grafické aplikace. Na rozdíl od předcházejícího vývojového prostředí TimeStorm, Qt Creator v sobě zahrnuje obrovskou nápovědu při vytváření aplikace. Je zde jak kompletní dokumentace všech dostupných funkcí, tak příklady nejrůznějších grafických aplikací pro lepší pochopení daného problému. Pohled na vývojové prostředí je obrázku (Obr. 23).

Obr. 23 Vývojové prostředí Qt Creator

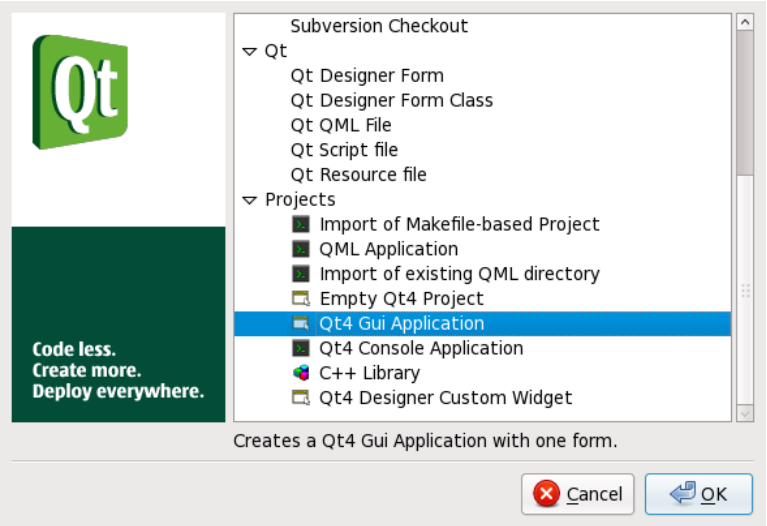
Na obrázku je vidět na levé straně panel se záložkami. Přepíná se zde mezi úpravou projektu (Edit), laděním (Debug), nastavením projektu (Projects), nápovědou (Help) a výstupem aplikace (Output). V levém dolním rohu jsou tři tlačítka, která slouží pro spuštění aplikace včetně její zkompilování, ladění nebo pouze zkompilování. V dolní liště jsou zase okna, která poskytují informace například o výsledku kompilování, zjištění chyb nebo ukazují výstup z aplikace.

Ve vývojovém prostředí lze vytvořit různé typy projektů, ať už jde pouze o obyčejný projekt bez grafického rozhraní nebo s grafickým rozhraním, nebo importovaný z jiného prostředí.

Pro vytvoření nového projektu pro architekturu ARM je opět nutné mít zkompilované grafické knihovny pro tuto architekturu a mít k dispozici i jejich kompilátor. Projekt je možno vytvořit i bez těchto knihoven, ale nebude jej možné spustit na cílovém zařízení. Tento způsob je

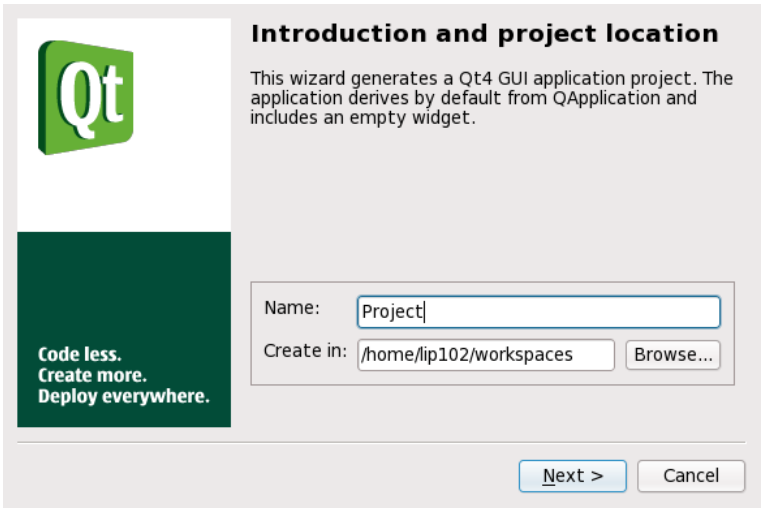
výhodný pro okamžité ladění bez nutnosti neustálého kopírování zkompilované aplikace do cílového zařízení. Proces vytváření projektu se takto mnohem urychlí.

Založení nového projektu se provede výběrem z nabídky menu *File -> New File or Project*. Objeví se nabídka s typem projektu. Je-li požadavek na aplikaci s grafickým rozhraním, jedná se o projekt **Qt4 Gui Application**. Ukázka je na obrázku pod textem (Obr. 24).



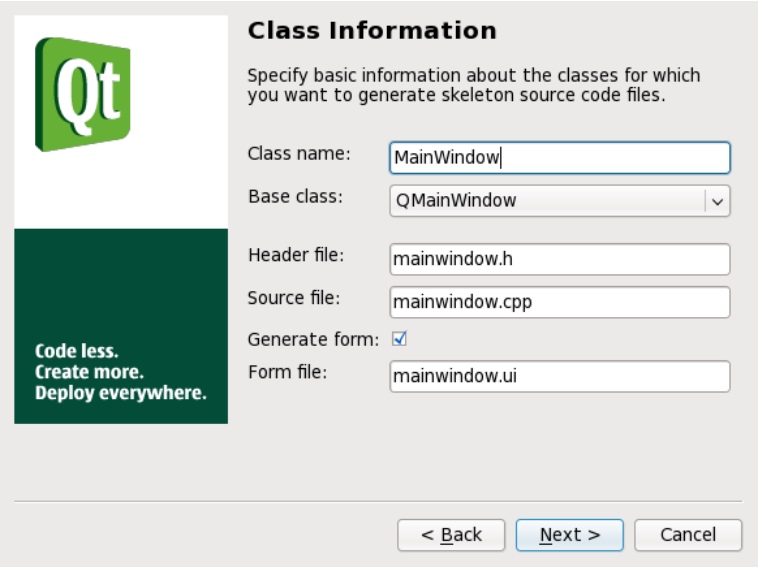
Obr. 24 Vytvoření nového projektu

Po vybrání se objeví obrazovka pro zvolení jména a místa uložení. Tuto situaci popisuje další obrázek (Obr. 25).



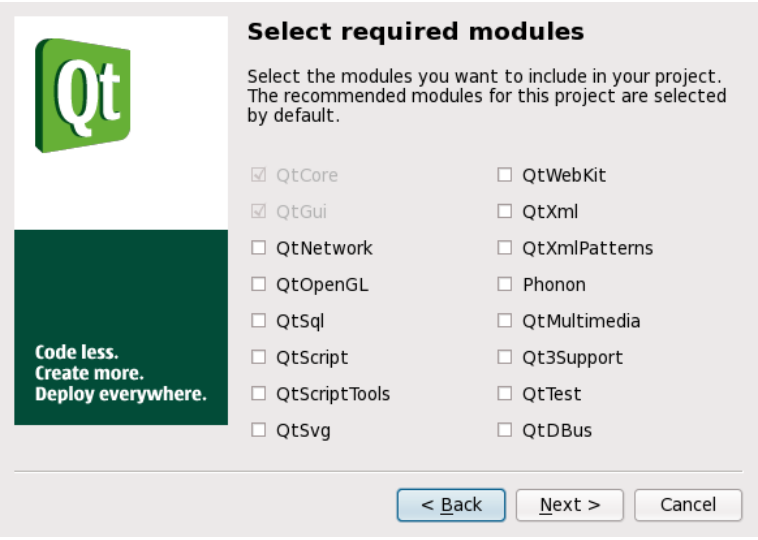
Obr. 25 Parametry projektu

Objeví se další obrazovka (Obr. 26), teď s nastavením názvu hlavní třídy grafického okna, hlavičkového souboru a zdrojového soubory. Není potřeba změny těchto údajů.



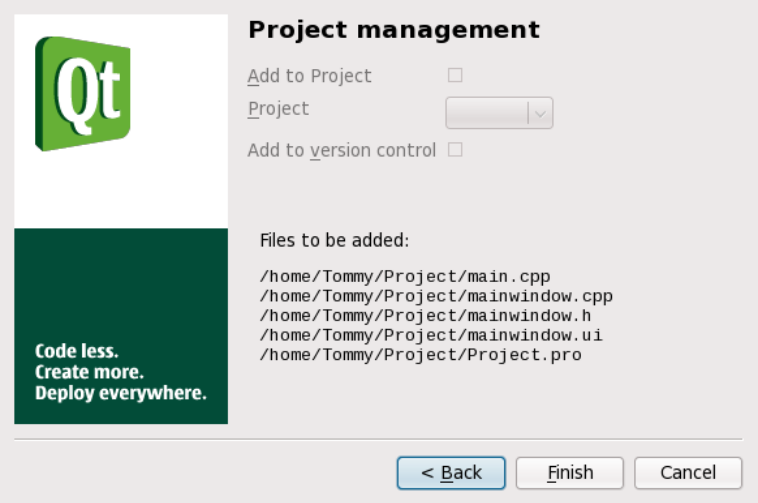
Obr. 26 Vytvoření nového projektu

Důležitým parametrem je výběr jednotlivých modulů. Pokud se jedná o aplikaci s grafickým rozraním, jsou automaticky zahrnuty moduly QtCore a QtGui. Ostatní moduly jsou pouze na výběru programátora. Nastavení projektu lze kdykoli změnit. Jsou zde moduly pro práci s webem, sítí nebo pro využití funkcí starší verze knihovny Qt3. Nastavení je na obrázku (Obr. 27).



Obr. 27 Výběr modulů

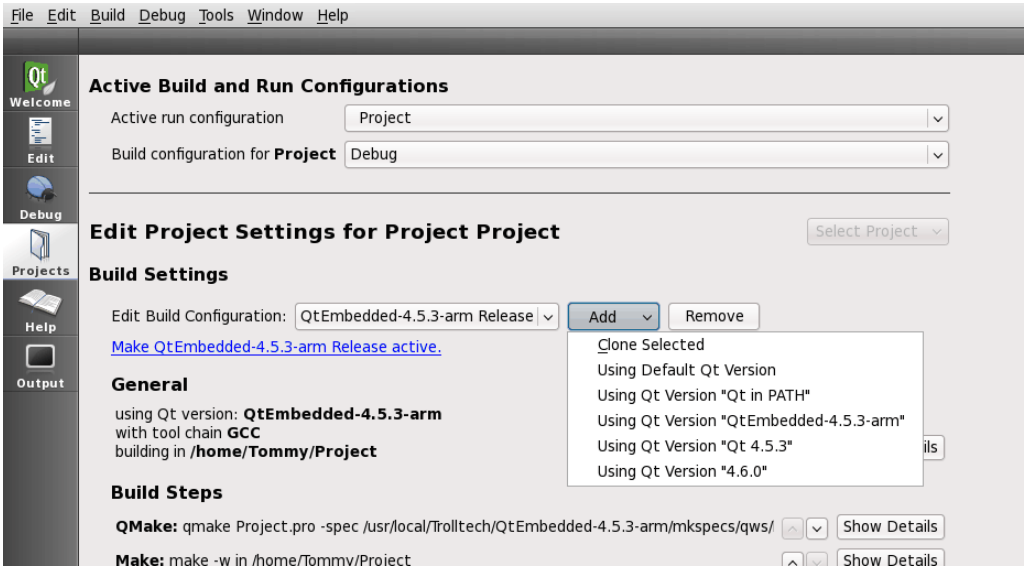
V posledním okně je pouze výpis vytvořených souborů.



Obr. 28 Dokončení projektu

Poslední věcí, kterou je nutné v projektu nastavit je kompilátor. Toto nastavení je levé záložce *Projects*. Výběr jednotlivých konfigurací se provede kliknutím na tlačítko *Add* v *Build Settings* a vybere se příslušná verze. Mezi těmito konfiguracemi lze libovolně přepínat (Obr. 29).

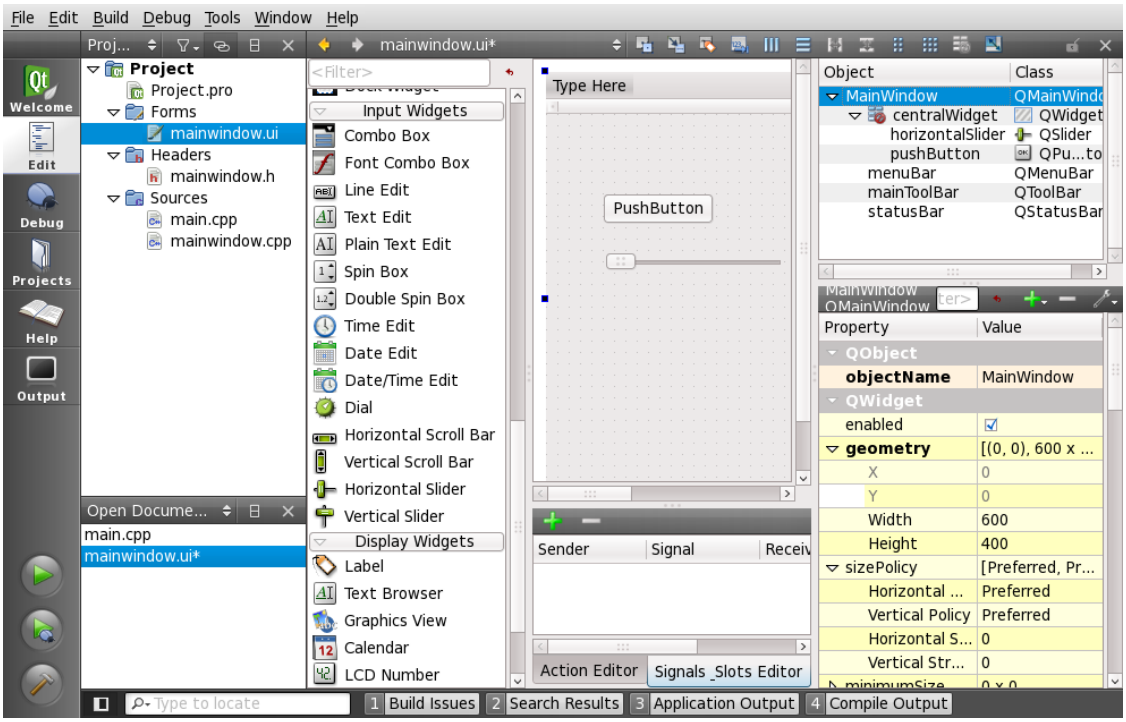
Postupy pro zkompilování grafické knihovny, včetně instalace kompilátoru a přidání podpory do vývojového prostředí jsou uvedeny v dokumentu v příloze (Příloha V).



Obr. 29 Výběr kompilátoru

Struktura vytvořeného projektu je na obrázku (Obr. 30). Součásti projektu je soubor mainwindow.ui, ve kterém jsou uloženy všechny informace o grafickém okně. Pro práci s tímto oknem složí nabídka grafických prvků, na obrázku je to druhý sloupec. Uprostřed obrázku je pohled na grafické okno a na pravé straně je okno pro úpravu vložených grafických prvků včetně jejich struktury.

Hlavičkový soubor má příponu .h a jsou v něm uloženy informace o použitých funkcích (např. stisk tlačítka). Soubor main.cpp je hlavní funkce programu a mainwindow.cpp je hlavní funkce grafického okna.



Obr. 30 Struktura nového projektu

9 Komunikace s robotickým zařízením

Dalším důležitým bodem je výběr vhodného způsobu komunikace. Existuje několik typů bezdrátových komunikačních technologií, mezi nejznámější patří například Bluetooth, WiFi, GSM nebo ZigBee. Každá z těchto technologií má jiné parametry a hodí se pro jiné použití. Nejpoužívanější jsou asi GSM a WiFi. GSM se používá v oblasti mobilních technologiích a váže se na služby mobilního poskytovatele, tudíž se nejedná o bezplatnou službu. Oproti tomu WiFi využívá bezlicenční pásmo 2.4 GHz, a lze ji tedy využívat zcela zdarma. Obě tyto technologie umožňují přenos velkých objemů dat. Mluví-li se o přenosové rychlosti a spolehlivosti přenosu WiFi, tyto parametry se značně mění s vlastnostmi prostředí. Na volném prostranství dosahuje až rychlosti několika desítek Mbit/s. Maximální rychlost je však 54 Mbit/s. WiFi má tu výhodu, že pokud je přítomno několik vysílačů, lze podle ní lokalizovat polohu. [10]

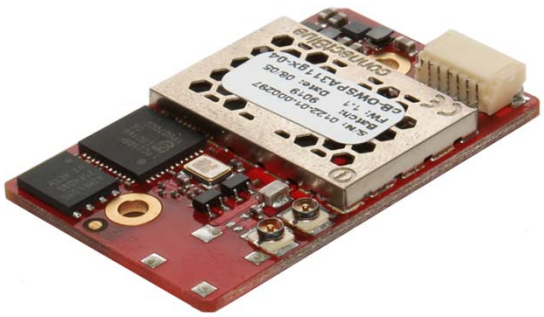
Bluetooth a ZigBee lze použít jak pro komerční, tak pro průmyslové použití. Dosahují malých přenosových rychlostí, podobně jako u WiFi využívají bezlicenční pásmo. Hodí se spíše pro menší vzdálenosti (do 100 m).

Byly uváženy jednotlivé komunikační možnosti a byla zvolena komunikace pomocí WiFi, protože dosahuje velkých přenosových rychlostí a použití je zdarma.

Jelikož i.MX31 nemá periferii umožňující komunikaci pomocí WiFi, bylo nutné tuto periferii dodat a k desce připojit. Byly použity komunikační moduly OWSPA311g. [9]

Vlastnosti WiFi modulu:

- Rychlost přenosu dat po sériovém rozhraní až 2,7 Mbit/s
- Podpora AT příkazů
- Napájení 3,3 – 5,5 V
- Zabezpečení WEP64, WEP128, WPA-PSK, WPA-PSK2
- Dosah s interní anténou 450 metrů
- WiFi 802,11b+g (54 Mbit/s)



Obr. 31 WiFi modul OWSPA311g

9.1 Oživení a nastavení WiFi modulu

Wifi moduly bylo nutné nejprve oživit a poté nakonfigurovat. Modul je připojen k i.MX31 přes sériové rozhraní UART, v Linuxu tomu odpovídá zařízení `/dev/ttymxc4` (na rozšiřující desce). Napájení WiFi modulu poskytuje sběrnice USB, která se nachází také na rozšiřující desce. Komunikační signály Rx (přijímací) a Tx (vysílací) jsou v CMOS logických úrovních ($-0{,}3\ V < V_L < 0{,}8\ V$, $2 < V_H < 3{,}3\ V$). Existuje několik způsobů připojení k modulu, nejjednodušší z nich je přes 6 pinový konektor (další z nich jsou v podobě board-to-board konektorů, kde se deska pouze přitiskne na piny druhého konektoru).

WiFi modul v podstatě funguje jako bezdrátová sériová linka a z pohledu uživatele se jeví jako přímé spojení s druhým modulem. Z jedné strany do něj vstupují data po sériovém rozhraní, ta jsou převedena a odeslána bezdrátově do druhého modulu, kde jsou přijata, převedena zpět a odeslána na sériové rozhraní.

Modul OWSPA311 pracuje ve dvou režimech. První z nich slouží pro jeho nastavení – režim AT příkazů. Druhý z nich je datový režim, ve kterém se modul nachází po připojení napájení a probíhá v něm samotná komunikace. Mezi režimy lze libovolně přepínat použitím únikové sekvence (escape sequence). Jedná se o určitou kombinaci znaků, která by se normálně v komunikaci neměla vyskytnout. Před a po použití sekvence se očekává, že modul nebude určitou dobu vysílat ani přijímat. [9]

Co se týče úspory napájení, disponuje modul třemi operačními režimy (Online, Sleep, Stop). V online režimu odebírá maximální proud 145 mA a komunikace probíhá neustále. Ve spacím (sleep) režimu modul šetří svou energie a vzbudí se pouze v případě, pokud chce vysílat nebo přijímat. Tento režim je nastaven jako výchozí. V režimu stop modul pouze udržuje spojení s přístupovým bodem (access point). [9]

Pro komunikaci mezi moduly bylo vybráno komunikační spojení AD-HOC, kde jeden modul pracuje jako server a druhý jako klient. Aby bylo možné se v budoucnu připojit několika moduly k vozidlu, bylo rozhodnuto, že server bude vozidlo a vzdálený řídicí systém klient.

Pro spojení dvou modulů, bylo nutné správně nakonfigurovat v AT módu server a klienta.

Nastavení serveru (vozidla):

- Nastavení sériové linky: rychlost 57600, 8 bitů, bez parity, 1 stop bit
- Vypnuté hardwarové řízení toku dat
- Typ sítě: AD-HOC
- Kanál: 9
- SSID WiFi sítě: D400_Car
- Pevná IP adresa 10.0.0.1, maska 255.255.255.0, brána 10.0.0.1
- Šifrování WEB 128 bitů, klíč: JiriKotzianXX
- Na portu 5003 spuštěno TCP poslouchání pro spojení s dalším modulem

Nastavení klienta (vzdálené řízení):

- Nastavení sériové linky: rychlost 57600, 8 bitů, bez parity, 1 stop bit
- Vypnuté hardwarové řízení toku dat
- Typ sítě: AD-HOC
- Kanál: 9
- SSID WiFi sítě: D400_Car
- Pevná IP adresa 10.0.0.2, maska 255.255.255.0, brána 10.0.0.1
- Šifrování WEB 128 bitů, klíč: JiriKotzianXX
- Na portu 5003 vypnuto TCP poslouchání
- Nastavení vzdálené osoby (remote peer) tcp://10.0.0.1:5003

Jednoduše lze říci, že Modul s IP adresou 10.0.0.2 se připojí ke druhému modulu s IP adresou 10.0.0.1 pomocí TCP spojení, které je šifrované s heslem „JiriKotzianXX“. Rozdíl mezi nastavením klienta a serveru je v povolení TCP poslouchání (server) a v nastavení remote peer (klient). Jakmile dojde ke spojení, začnou moduly spolu komunikovat a posílají si vzájemně data.

9.2 Komunikační protokol

Komunikační protokol ovládacího panelu je totožný s komunikačním protokolem průzkumného vozidla. V rámci této práce byl pouze upraven.

Pro přenos velkého objemu dat je nutné vytvořit vlastní komunikační protokol, aby bylo jasné, o jaký typ dat se jedná, co bude které položka obsahovat a jaká hodnota se bude kde nacházet. Komunikační protokol by měl být jednoduchý. Bylo rozhodnuto, že data se budou přenášet binárně a ne textově. Požadavek by i na ignorování AT příkazů pro WiFi modul a tím vyhnutí se případných problémů s jejich náhodnou konfigurací.

Struktura komunikačního protokolu:

```
STX LLH LLL TYP DATA EXT
```

Začátek zprávy je označen bytem STX (0x02), následuje délka přenášené zprávy. Délka může být maximálně 65535 bytů. Délka zahrnuje i jeden byte, který definuje typ zprávy, ten je označen TYP. Je zřejmé, že protokol rozlišuje 256 druhů zpráv. Za typem již následují binární data a ta jsou ukončena ukončovacím znakem ETX (0x03).

V souboru *global.h* jsou definovány jednotlivé zprávy, které se budou vysílat a přijímat (velikost, fyzikální rozsah, směr). O komunikaci se stará komunikační proces *vehicle_wifi*. Ten zajišťuje přenesení dat ze sdílené paměti do vzdáleného ovládání a naopak. Ostatní procesy se už nestarají o komunikaci, pouze zapisují a čtou data ve sdílené paměti. Díky tomu je komunikace zcela oddělena od dalších úloh a procesů a je na řízení do velké míry nezávislá.

Sériová linka a komunikační protokol neřeší potvrzení zpráv, tak musely být přidány dvě speciální zprávy. Zpráva *SENDISREADY* se vysílá v každém cyklu jako první a povinností druhé strany je odpovědět do dvou sekund zprávou *RECEIVEISREADY*. Pokud je přijata tato zpráva, komunikační médium je v pořádku a druhé zařízení je připojeno. Toto je zvláště důležité při manuálním řízení, neboť výpadek komunikace při jízdě by mohl znemožnit například zastavení. Vozidlo tedy na stav těchto zpráv musí reagovat.

9.3 Výsledky použití WiFi modulu OWSPA311

I když se zdá, že komunikace pomocí WiFi je dobrým řešení, tak s použitím modulu OWSPA311 vznikaly problémy s komunikací. Výrobce dokonce zaručuje dosah signálu na volném prostranství až 400 metrů s externí anténou.

Při testování komunikace vznikaly problémy na obou stranách v případě přijetí dat a to i na velmi krátké vzdálenosti jako jednotky metrů. Z počátku se zdálo, že spojení nefunguje správně a moduly jsou špatně nakonfigurovány. Po hlubší analýze se došlo k závěru, že opravdu všechna data dorazila v pořádku ale se zpožděním. Zpoždění dosahovalo místy až několika sekund. Takové zpoždění však nejde akceptovat, jde-li o manuální řízení vozidla. Proto byly implementovány do vozidla algoritmy na ochranu při výpadku komunikace na delší dobu. Nakonec byla optimalizována komunikace jak na straně vozidla, tak vzdáleného řídicího systému. Komunikační protokol byl redukován a data byla odesílána tak, aby se rovnoměrně zatížila sběrnice a snížila se tak celková přenosová rychlost.

WiFi moduly byly nakonfigurovány, aby se dosáhlo maximálního výkonu. Byly přepnuty do režimu online, tedy nepřetržitého chodu, byla vypnuta úspora energie. Rychlost sériové komunikace s modulem byla rapidně snížena z původního 460800 Baud/s na 57600 Baud/s. Podobně byla i snížena rychlost bezdrátové komunikace.

I přes všechna tato opatření problémy s komunikací přetrvávají. Bezdrátová komunikace WiFi je sice dobrým řešením, ale s použitím modulů OWSPA311 se moc neosvědčila.

10 Ovládací panel

Ovládací panel je grafická aplikace, která bude poskytovat uživateli graficko-uživatelské rozhraní. Prostřednictvím ní bude uživatel získávat informace o vozidle a bude mít také možnost zadávat mu určité příkazy. Součástí diplomové práce je tuto aplikaci navrhnout a naprogramovat.

Tato kapitola nastíní návrh aplikace ovládacího panelu, včetně funkcí které bude poskytovat uživateli. Grafikou realizací se bude zabývat další kapitola.

10.1 Návrh ovládacího panelu

Ovládací panel umožňuje uživateli (operátorovi) vzdálenou obsluhu vozidla. Před samotným grafickým návrhem, je nutné si ujasnit seznam požadavků a funkcí, které bude panel poskytovat.

Hlavní funkce ovládacího panelu

- Komunikace se vzdáleným robotickým zařízením (vozidlem)
- Poskytování přehledné vizualizace údajů o vozidle
- Možnost zobrazení historických údajů ve formě grafu
- Nastavení počáteční a cílové pozice na mapě
- Pohyb vozidla bude zaznamenáván na mapě
- Přehledné a snadné ovládání na dotykové obrazovce
- Možnost manuálního řízení vozidla na dálku

Základní parametry panelu

- Rozhraní mezi operátorem a aplikací tvoří dotyková fólie
- Syntaxe programovacího jazyka C++
- Aplikace využívá grafickou knihovnu Qt
- Ladění přes sériové rozhraní nebo ethernet
- Manuální řízení umožňuje joystick připojený k USB portu
- Aplikace bude uložena na paměťové SD kartě

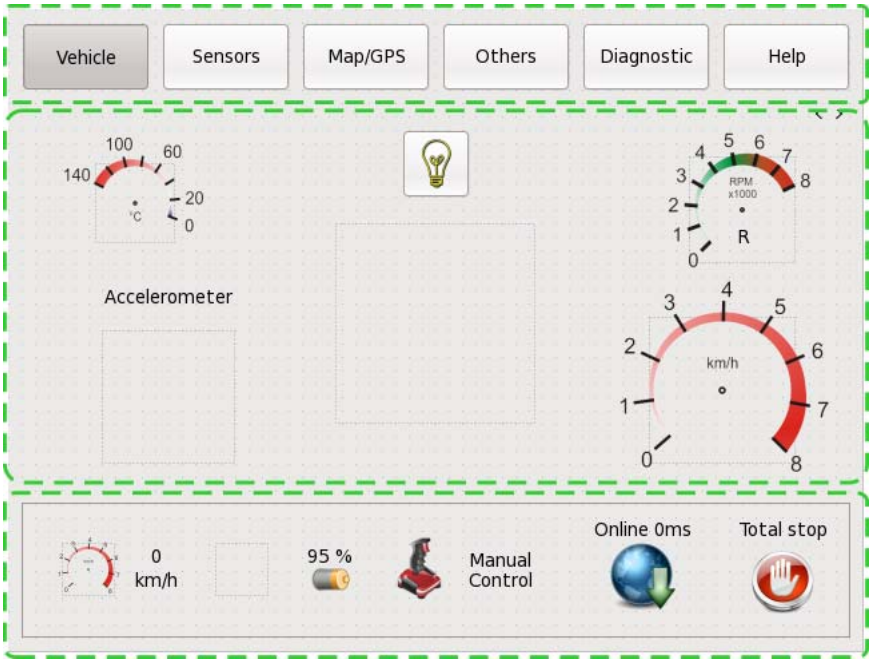
Ovládací panel má tedy za úkol zobrazovat veškeré odeslané údaje z řídicí jednotky o sobě a ostatních modulech umístěných na vozidle, a má mít možnost ovládat robotické zařízení na dálku.

Co se týče grafické realizace, bude ovládací panel rozdělen do několika obrazovek, protože vozidlo poskytuje velké množství dat a nelze je všechny najednou zobrazit. Každá obrazovka bude zaměřena na určitý problém – informace o vozidle, mapy, senzory a další. Aplikace musí být přizpůsobena pro práci s dotykovou obrazovkou.

10.2 Realizace ovládacího panelu

Jak již bylo řečeno v návrhu, mezi řídicí jednotkou vozidla a vzdáleným řídicím systémem se přenáší velké množství dat. Tato data byla rozdělena podle funkce a setříděna, aby například společné věci jako senzory byly s ostatními senzory. Samozřejmě i zde existují data, která by měl mít uživatel stále na očích a nemusel je složitě nelézat a neustále mezi nimi přepínat.

Panel byl proto pomyslně rozdělen na tři části. První část (horní) slouží pro přepínání mezi okny, druhá (prostřední) ukazuje jednotlivá okna (stránky) a poslední, třetí část (dole) zobrazuje globální data. Rozdělení hlavního okna znázorňuje (Obr. 32).



Obr. 32 Obrazovka informací o vozidle

Do globálních dat patří přepínání mezi ručním a automatickým režimem, kontrola stavu baterie vozidla, stav komunikace, nouzové zastavení, ukazatel rychlosti a natočení kol vozidla. Funkce nouzové zastavení a přepínání režimů řízení jsou konstruována jako tlačítka.

Bylo rozhodnuto, že popisky a texty budou psány v anglickém jazyce. Samotná aplikace byla vytvořena, aby umožňovala přepínání mezi češtinou a angličtinou, ale jako výchozí jazyk při navrhování byla vybrána angličtina.

Není způsob jak zobrazit všechny položky najednou v jednom okně, tak byl panel rozdělen do několika oken pomocí **stackedwidget** grafického prvku. Je jednodušší přepínat mezi widgety (prvky uživatelského rozhraní), které jsou rozděleny na stránky, než vytvářet pro každou skupinu nové okno. Pro přepínání mezi stránkami slouží tlačítka na horní straně obrazovky. Jsou velká, aby se snadno dala ovládat přes dotykovou obrazovku.

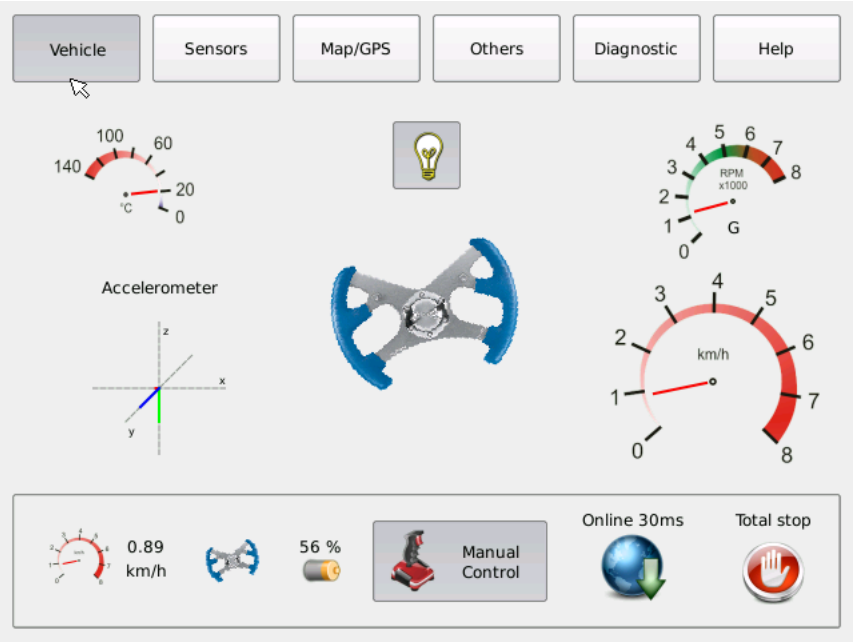
Odporová fólie funguje na principu změny odporu v místě dotyku. Nedá se říci, že jde o převratnou technologii, protože stejné změny odporu lze docílit stiskem např. v jiných částech obrazovky. Existují různé filtry, které dokážou dosáhnout částečné eliminace šumu, který při stisku vzniká. Obvyklý princip činnosti tlačítka je založen na jeho stisku a poté uvolnění, následně se vyvolá patřičná událost. Důležité je, aby kurzor v době stisku a uvolnění byl stále nad tlačítkem. Občas vznikl problém, kdy tlačítko sice stisknuto bylo, ale při jeho uvolnění, se vlivem nedokonalosti odporové fólie přemístil kurzor na jiné místo a nemohla být tak vyvolána událost. Proto všechna použitá tlačítka ve vizualizaci reagují pouze na prvotní stisk a nečekají na uvolnění.

Byla zde zvážena i možnost nahrazení celého systému záložkami. Tento způsob se ale neosvědčil, protože není možné nastavit ručně rozměr pro oblast na přepínání (místo, kde je nutné kliknout, aby se přepnula záložka). Tak se přešlo na metodu samostatných velkých tlačítek ve spojení s prvkem **stackedwidget**.

Pro jednodušší představu budou jednotlivé obrazovky zobrazeny přímo z vývojového prostředí, aby bylo jasné rozložení grafických prvků. Poté budou případně vyobrazeny jako výstup aplikace, bude-li to nutné. Oba obrázky mají jiné zbarvení, protože dotykový panel pracuje pouze s 262 tisíci barvami.

10.2.1 Informace o vozidle

První obrazovka jsou informace o vozidle. Poskytují uživateli údaje o rychlosti, otáčkách, natočení kol, zapnutí světel a akcelerometru. Demonstrace je na obrázku (Obr. 33).



Obr. 33 Obrazovka informací o vozidle (výstup)

Většina grafických efektů, jako jsou ručičky otáčkoměrů, grafy nebo mapy, byly vytvořeny přes funkce primitivního malování. To umožňuje vytváření jakýchkoli grafických objektů, jako jsou body, čáry, obdélníky, elipsy a další složitější tvary.

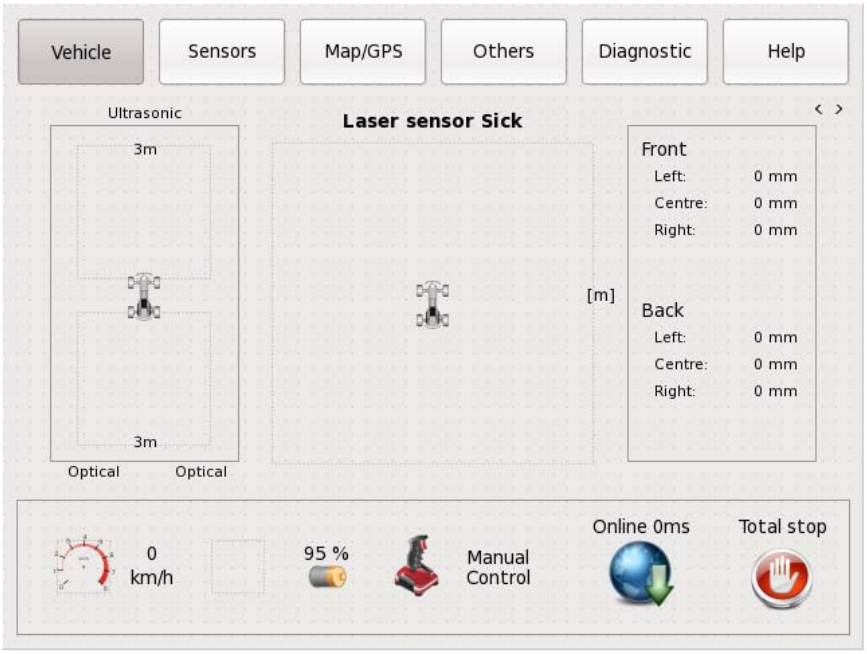
Princip činnosti obyčejného otáčkoměru byl založen na jednoduché animaci. Grafický prvek vykresluje pouze přímku otáčející se kolem jednoho bodu (většinou počátek souřadnicového systému) a nad ním je umístěn obrázek budíku. Podobně je tak vytvořena představa otáčejícího se volantu. Na rozdíl od přímky je zde obrázek, který se opět otáčí kolem jednoho bodu. Úhel natočení obrázku volantu je dán hodnotou skutečného natočení vozidla. Co se týče úhlu natočení ručiček otáčkoměrů, musely být přepočítány, aby hodnoty seděly s hodnotami na obrázcích budíků.

Srovnáním obrázků (Obr. 32) a (Obr. 33) je vidět, které prvky musely byly vytvořeny přes funkce primitivního malování.

10.2.2 Senzory

Obrazovka senzorů dává uživateli představu o dostupnosti terénu a možných překážkách před i za vozidlem. Přepnutí na tuto stránku je možné opět klinutím na horní panel tlačítek.

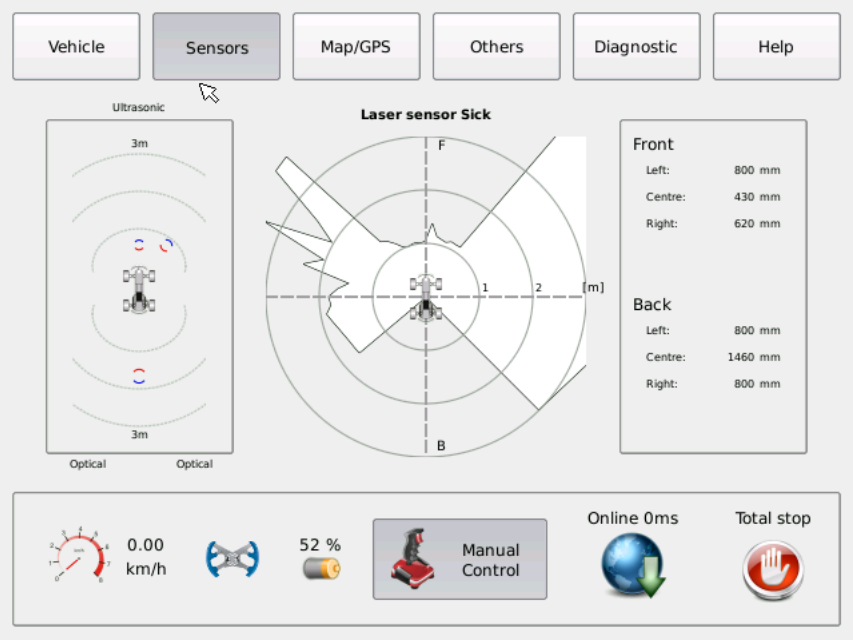
Na vozidle je umístěn laserový senzor, jehož výstupem je obzor o úhlu 270°. Prostor o velikosti 270° je rozdělen po 5° na 54 hodnot, kde každá hodnota udává vzdálenost od překážky (maximální dosah je 20 metrů). Hodnoty jsou dále přepočítány do kartézského systému souřadnic (x, y). Pro vyznačení oblasti změřené senzorem, byl pro jednoduchost použit obyčejný polygon složený z 54 bodů. Vnitřní plocha, tedy prostor bez překážky, je vybarvena bílou barvou, okraje černou a vnější plocha naopak barvou pozadí (průhledná barva).



Obr. 34 Obrazovka senzorů

Vozidlo je vybaveno navíc analogovými dálkoměry a ultrazvukovými senzory. Těm je věnována levá a pravá strana obrazovky. Levá strana poskytuje celkový náhled na vozidlo a grafické znázornění výskytu překážky před i za ním do vzdálenosti maximálně 3 metrů. Numerické znázornění hodnot naměřených senzorů je na pravé straně.

Na obrázek z vývojového prostředí (Obr. 34) je možno vidět pouze prázdné plochy, které jsou použity pro vykreslení. Výsledek aplikace je na obrázku (Obr. 35).



Obr. 35 Obrazovka senzorů (výstup)

Plocha, na kterou je vykreslován polygon, byla ocejchovaná stupnicí pro lepší přehlednost, podobně tak i obrázek nalevo. Navíc velikost polygonu, stupnice a obrázek vozidla se dynamicky mění podle rychlosti vozidla. Bylo řečeno, že senzor dokáže zaznamenat překážku vzdálenou až 20 metrů. Pokud by byla vykreslena celá plocha do 20 metrů, obrázek by byl příliš malý na to, aby se dalo pohodlně manévrovat vozidlem při nízkých rychlostech. Proto je celý obrázek dynamicky zmenšován a zvětšován. Například pro nulové rychlosti je dosah viditelné mapy pouze do 3 metrů, naopak při nejvyšších rychlostech až 20 metrů

10.2.3 Grafické operace

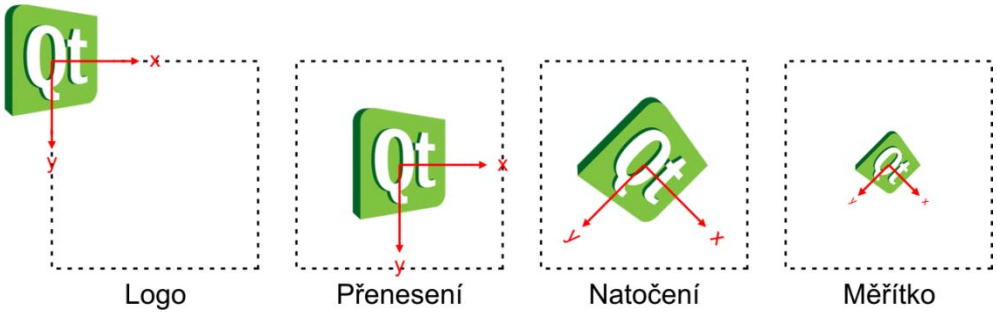
Předtím než bude popsán návrh obrazovky pro vykreslování map, je nutné něco říci o základních grafických operacích s Qt knihovnou.

QPainter je třída, která umožňuje nízko úrovňové kreslení na widgetech nebo jiných zařízeních. Poskytuje také vysoce optimalizované funkce pro kreslení GUI. Je možné vykreslit naprosto cokoli, od jednoduchých čar po komplexní tvary jako koláč nebo tětivu. Pracuje také s obrázky a texty. Vykresluje do kartézského souřadnicového systému. [5]

Plocha, do které je obrázek vykreslen, není u nízko úrovňového malování přesně definována jako kreslící plátno, protože plátnem může být v podstatě cokoli. Pomocí třídy QPainter je možné vykreslit například čáru do obyčejného tlačítka nebo jiného grafického prvku.

Základem je vytvoření nového objektu v nové třídě, vycházející z obecné třídy widget. Do ní je nutné přidat funkci **paintEvent**, která slouží pro vykreslování. Poté u vytvořeného objektu se zavoláním funkce **update** automaticky vyvolá událost **paintEvent**, ve které už se provádí samotné vykreslování. Velikost vzniklého "kreslícího plátna" je dána pevnými rozměry vytvořeného objektu.

Obrázek (Obr. 36) popisuje některé ze základních operací ve třídě QPainter. Třída využívá několik systémů souřadnic (stále jde o kartézské souřadnice). Výchozí počátek souřadnic je vždy v levém horním rohu.



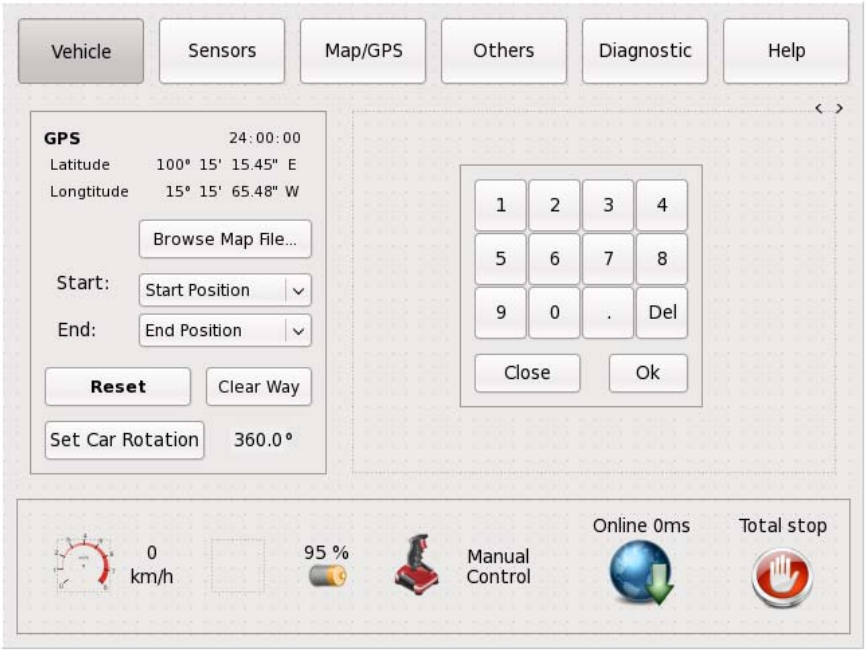
Obr. 36 Základní grafické operace

Z prvního obrázku by se dalo říci, že šedý obdélník a logo Qt používají stejný souřadnicový systém. Ve druhém obrázku je logo posunuto do středu obdélníku. V podstatě došlo k tomu, že byl oddělen souřadnicový systém obdélníka a obrázku loga, a počátek souřadnicového systému loga Qt byl posunut do středu obdélníku (funkce **translate**). Hodnoty x a y jsou ve všech případech stejné (0, 0). K manipulaci souřadnicových systému slouží funkce **save** a **restore**. U první funkce se jedná o tzv. uložení současného stavu rozkresleného obrázku, kdy je nakreslen pouze šedý obdélník, poté se provede vykreslení loga s osami a popisky. Voláním funkce **restore** dojde k obnovení původního obrázku. Výsledkem jsou dva nezávislé obrázky vykreslené do jednoho okna. Tím jde docílit například posouváním pouze určité skupiny objektů, jejich otáčení (**rotate**) nebo dosáhnout snadné směny velikosti (**scale**).

Tímto způsobem lze provádět různé grafické operace na různých objektech, aniž by to ovlivňovalo ostatní objekty.

10.2.4 Mapy a GPS navigace

GPS souřadnice a zobrazení vozidla na mapě umožňuje třetí obrazovka (Obr. 37). Na levé straně jsou funkce pro práci s mapou (načtení konkrétní mapy ze souboru, nastavení počáteční a cílové pozice podle daných bodů nebo ručně vlastním výběrem) a také údaje s GPS na vozidle (čas a souřadnice). Soubor, ze kterého je načtena mapa, musí být dostupný v systému souborů. Jsou v něm obsaženy souřadnice objektů, jako jsou zdi, dveře a trasa. Z nich se poté vykresluje celá mapa. Pokud není soubor načten, je mapa prázdná. Na pravé straně byla vytvořena velká oblast, do níž se celá mapa vykresluje.



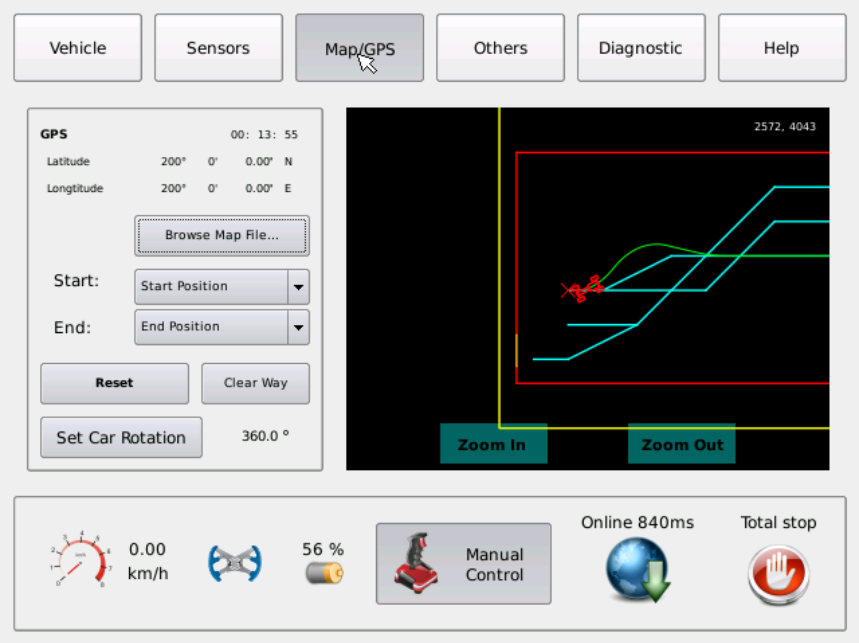
Obr. 37 Obrazovka map a navigace

Pro ruční zadávání výchozího natočení vozidla bylo vytvořeno speciální okno s numerickou klávesnicí, to proto, že posuvníky nebo jiné widgety se moc neosvědčily. Je to jediné místo, kde byla potřeba numerické klávesnice.

Nejzajímavější částí bylo vytvoření okna pro vykreslování mapy, trasy, vozidla a panelu pro ovládání do jednoho grafického okna. Použity byly celkem tři systémy souřadnic. Jeden z nich byl použit pro vykreslení ovládacího menu pro manipulaci s mapou. Poskytuje funkce pro posunutí, přiblížení nebo oddálení, nastavení startovní a cílové pozice. Druhý souřadnicový systém vykresluje mapu a historii trasy vozidla. Umožňuje dynamické posunutí mapy, její přiblížení a oddálení podle potřeb uživatele. Třetí souřadnicový systém vykresluje obrázek vozidla a natáčí jej podle skutečného natočení vozidla na mapě.

Výsledkem je obrázek autíčka ve středu obrazovky, které se otáčí kolem svého středu. Pod ním je vykreslena mapa, která se automaticky posouvá a dělá tak představu pohybujícího se autíčka

po mapě. Případně po okrajích obrazovky je vidět ovládací menu. Je-li aktivní ovládací menu, je vypnuta funkce automatického posunu mapy, aby bylo možné nastavit ručně startovní a cílovou pozici. Ukázka obrazovky s mapou je na obrázku (Obr. 38).



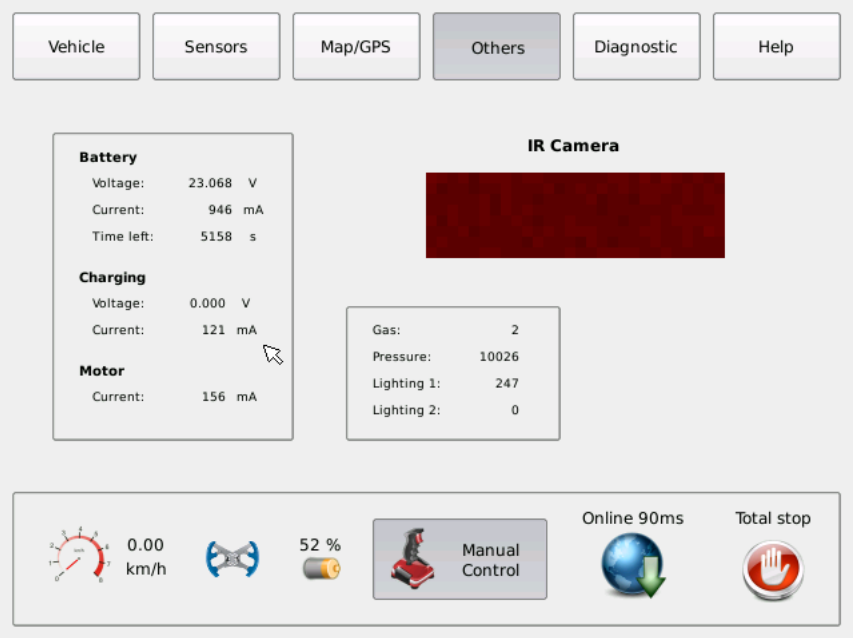
Obr. 38 Obrazovka map a navigace (výstup)

Ruční posun mapy a nastavení startovní nebo cílové pozice je možný také dotykem ukazovátka (prstem) po dotykové obrazovce na grafickém okně s mapou. Prvním kontaktem se zobrazí ovládací menu a dalším dotykem po menu se aktivuje posun. Dotyk mimo ovládací menu, menu deaktivuje.

10.2.5 Ostatní senzory

Zbylé senzory jsou zařazeny v dalším okně, jelikož nemají s navigací ani mapou nic společného. Jsou zde uloženy informace o napětích a proudech baterie i motoru a ostatních čidlech. Není potřeba dlouhého popisování, protože převážná většina hodnot je vyjádřena čísly.

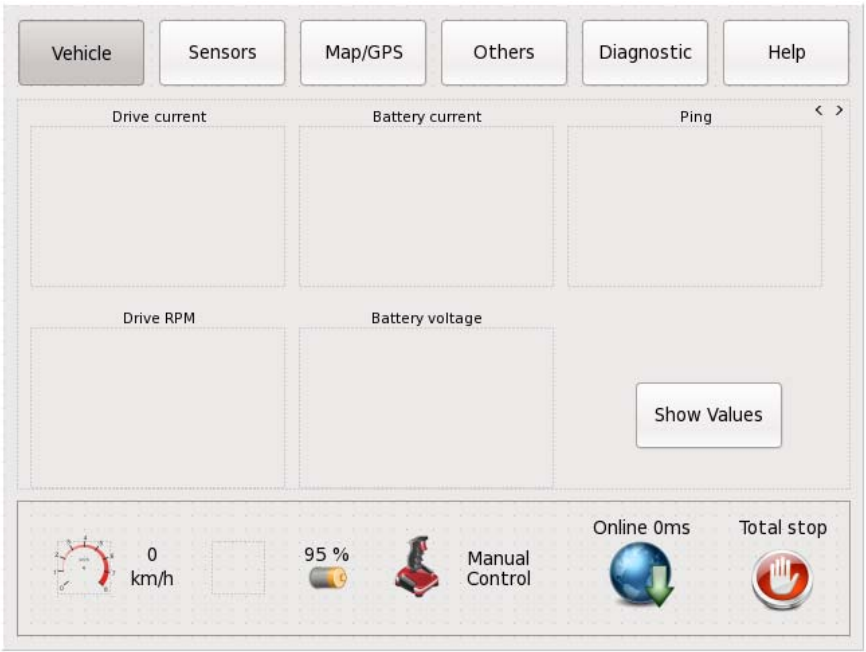
Na vozidle je upevněna na servo motoru infračervená kamera (IR). Otáčí se do 31 poloh a v každé poloze poskytne 8 hodnot ve sloupci o teplotě před sebou. V ovládacím panelu bylo toho prezentováno obrázkem 31 x 8 bodů. Obvykle má každý bod rozsah 256 hodnot odstínů červené (0 je černá, 255 je červená). Barva odpovídající teplotě lidského těla by vypadala skoro jako černá. Proto musela být závislost teploty na barvě aproximována logaritmickou křivkou. Tím se sice zmenšil maximální rozsah teplot, který je možno zobrazit, ale na druhou stranu lze snadněji rozlišit teploty okolo 50 °C. Ukázku demonstruje obrázek pod textem (Obr. 39).



Obr. 39 Obrazovka ostatních senzorů (výstup)

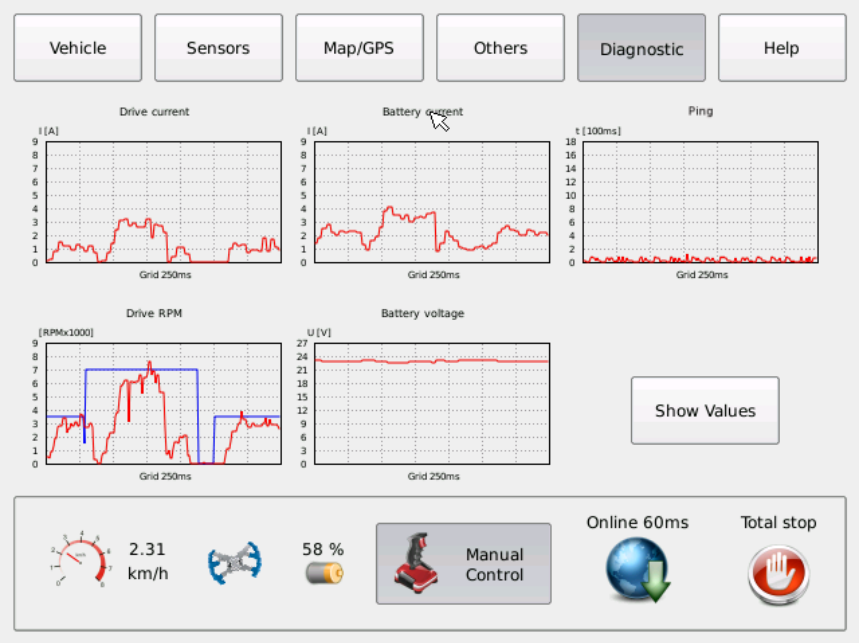
10.2.6 Diagnostika a Grafy

Další stránka na ovládacím panelu byla navržena tak, aby umožnila uživateli zobrazit historické údaje prostřednictvím grafů.



Obr. 40 Obrazovka grafů

Na obrázku (Obr. 40) je vidět 5 kreslících pláten, které zobrazují hodnoty napětí, proudů a odezvy komunikace v závislosti na čase. Do každého plátna je vykreslována jak samotná hodnota proměnné, tak mřížka a popisky os. V grafu pro otáčky jsou dokonce vykresleny dvě proměnné (požadované otáčky a aktuální otáčky). Každá vykreslená křivka má 175 hodnot. Při příchodu nové hodnoty, se všechny body ve křivce (kromě prvního) posunou o jednu pozici doleva a nová hodnota se zapíše do poslední buňky vpravo. Vytváří se tak představa pohybujícího se grafu. Výstup aplikace během jízdy vozidla je na obrázku (Obr. 41).



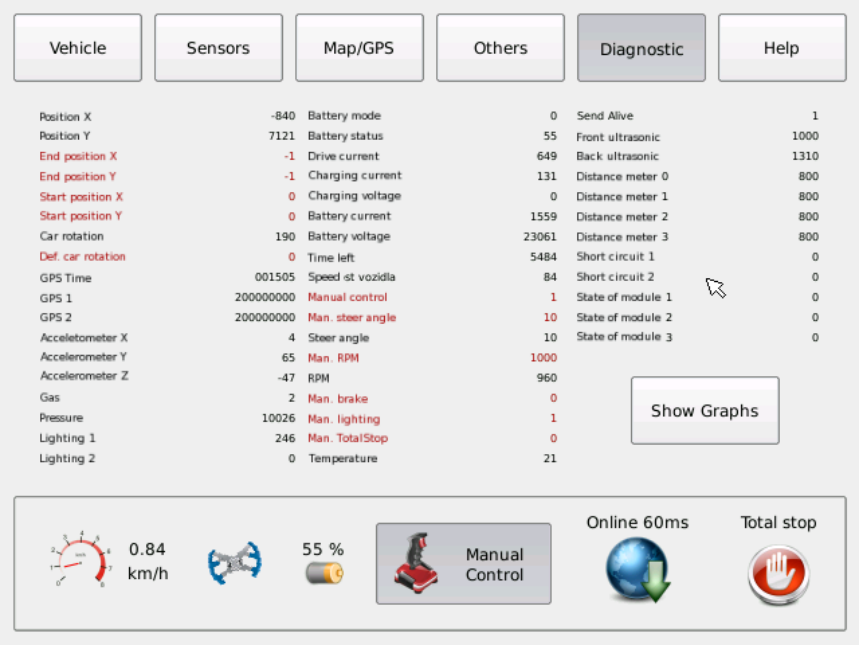
Obr. 41 Obrazovka grafů (výstup)

Kreslící prvek, který byl použit pro vykreslení křivky, se nazývá **drawpath**. Jak název napovídá, půjde o kreslení trasy. Ze 175 vložených bodů vytvoří spojitou čáru, kterou následně vykreslí do plátna. Tento způsob je mnohem jednodušší a méně náročnější na výkon, než vytváření přímek mezi každými dvěma body.

Aby měl uživatel kontrolu nad všemi naměřenými hodnotami najednou a nemusel listovat mezi ostatními stránkami (okny), je zde schováno ještě jedno okno. To poskytuje diagnostiku veškerých údajů, kromě hodnot z laserového senzoru a IR kamery (hodnot je příliš mnoho, aby byly zobrazeny společně s ostatními, navíc názornější by byla spíše grafická ukázka). Aktivuje se stiskem tlačítka vpravo dole v okně diagnostiky.

Nejpoužívanějším prvkem grafického uživatelského rozhraní je **label** (popisek). Dalo by se říci, že i nejjednodušší. I když se používá převážně jako popis textu, může být použit i pro vykreslení obrázků, které je možné vidět v obrazovce o informacích o vozidle (Obr. 32). V okně diagnostiky vypisuje hodnoty získané z komunikačního protokolu. Vždy se jedná o text a to i v případě, je-li zobrazeno číslo např. s desetinou čárkou.

Qt knihovna obsahuje funkce pro práci s textem, tudíž není problém převodu čísla, celého nebo s desetinnou čárkou, na prostý text a zpátky. Tento převod je nutný vždy, pokud má být číselná hodnota zobrazena přes popisek.



Obr. 42 Obrazovka diagnostiky (výstup)

Diagnostika poskytuje přehled surových dat (neupravené hodnoty) získané z komunikace s vozidlem. Surová data z důvodu lepšího ladění v případě výskytu chyby. Pohled na obrazovku diagnostiky je na obrázku (Obr. 42).

10.2.7 Nápověda

Posledním oknem je nápověda. Má poskytnout uživateli pomoc při manuálním řízení vozidla. Aby lépe pochopil, k čemu slouží jednotlivá tlačítka na joysticku. Zajímavostí je zde funkce pro sejmutí obrazovky každých 5 sekund. Obrázky jsou ukládány v souborovém systému do stejné složky, ve které je samotná aplikace, a jsou ve formátu png. Právě tyto obrázky byly použity pro v textu.

V úvodu bylo řečeno, že aplikace umožňuje přepínání mezi jazykem angličtinou a češtinou. K tomu byla použita dvě tlačítka **radiobutton**. V podstatě jde o přepsání všech textů do požadovaného jazyka.

Obr. 43 Obrazovka nápovědy

10.3 Optimalizace vykreslování

Vývojová sada i.MX31 LiteKit nemá v sobě žádné grafické jádro (GPU), které by se staralo o grafické operace, proto všechno musí zpracovávat sám procesor. Náročnější operace, jako jsou například animace, nebo neustálé překreslování grafických pláten, zaberou procesoru většinu času výpočetního výkonu, než samotné vykonávání programu. Čím větší je vykreslená plocha a čím častěji se překresluje, tím více času potřebuje procesor na její zpracování. Proto musely být některé grafické funkce optimalizovány, aby měl procesor čas i na ostatní výpočty.

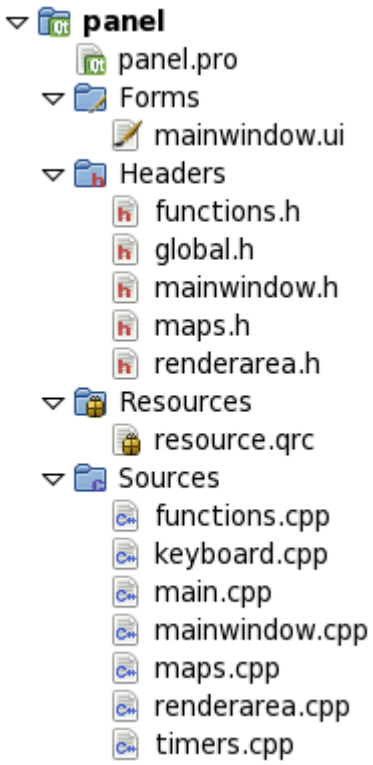
Údaje z vozidla jsou odesílány každých 100 ms, takže není potřeba překreslovat celou obrazovku s grafikou v menších intervalech. Z tohoto důvodu, se všechny grafické operace provádějí cyklicky každých 100 ms. Ušetřením času procesoru se dosáhlo i tím, že není nutné znovu překreslovat obrázek, pokud byla přijatá hodnota stejná, jako hodnota minulá. Týká se to zejména natočení volantu, ručiček otáčkoměrů nebo zakreslování překážek u optických senzorů a dálkoměrů.

Kromě již zmíněných optimalizací, funguje celá aplikace jako jeden velký přepínač. Podle toho, které okno (stránka) je zrovna aktivní, se vykonávají příslušné operace. Protože není nutné přepočítat mapu v okamžiku, kdy je spuštěno okno s nápovědou nebo senzory.

Použitím uvedených metod při návrhu aplikace došlo k výraznému snížení nároků na výpočetní výkon při vykreslování grafických efektů a jiných animací, použitých v ovládacím panelu.

10.4 Struktura programu ovládacího panelu

Pro lepší názornost bude použit obrázek z vývojového prostředí se seznamem použitých souborů. I když jsou soubory ve vývojovém prostředí vidět v jednotlivých složkách, ve skutečnosti tyto složky v projektu nejsou, je to pouze kvůli lepší přehlednosti.



Obr. 44 Struktura programu řídicí aplikace

Nastavení projektu je v souboru panel.pro. Je zde uložena struktura všech použitých souborů a informace o použitých modulech.

Forma, tedy grafická stránka aplikace je v souboru mainwindow.ui. Z této formy jsou při kompilaci vytvořeny další soubory, do kterých je vygenerován zdrojový kód grafiky. Projekt je tudíž možné vytvořit i z těchto vygenerovaných souborů a pak není potřeba formy mainwindow.ui.

Hlavičkové soubory obsahují většinou struktury a funkce použité v programu. Celá struktura komunikačního protokolu je uložena v souboru global.h. téměř stejná struktura je použita i v průzkumném vozidle.

V souboru functions.cpp a maps.cpp se nachází funkce pro práci s mapou. Dokážou z mapových podkladů získat data o souřadnicích objektů, a ta jsou následně použita pro vykreslení v grafickém okně. Jsou zde i funkce pro vyhledání nejbližšího vrcholu v mapě.

Funkce tlačítek numerické klávesnice, která byla použita u obrazovky mapy pro zadávání výchozího natočení vozidla, jsou v souboru keyboard.cpp. Ošetřuje i možnosti zadání špatného formátu dat.

Soubor main.cpp obsahuje hlavní funkci (main) programu, ze které je spouštěno grafické vlákno. Navíc je zde iniciace sdílené paměti a mutexu pro její zablokování.

Hlavní funkce grafické aplikace mainwindow je definována v souboru mainwindow.cpp. Načítá si soubory, které jsou vygenerovány z formy grafického okna a veškeré vytvořené hlavičkové soubory. Krom toho obsahuje například funkce pro přepínání obrazovek ovládacího panelu, přepínaní mezi angličtinou a češtinou. Z počátku vývoje byla velká většina aplikace obsažena v tomto souboru. Soubor byl ale dosti obsáhlý a proto byl rozdělen podle funkce do tří souborů - keyboad, timers a mainwindow.

Vlastní grafické prvky, které byly vytvořeny v rámci diplomové práce, jsou uloženy v souboru renderarea.cpp. Jsou zde definována všechna kreslící plátna (mapa, ručičky otáčkoměrů, obrazy ze senzorů, atd.).

Do posledního souboru timers.cpp byly umístěny funkce pro časovače. Tyto funkce byly vytvořeny, aby zajišťovaly periodickou aktualizaci hodnot grafických prvků a vykreslování objektů grafických pláten, resetovací sekvenci pro nastavení startovní a cílové pozice vozidla a uložení celé obrazovky jako obrázek. Princip vykonávání programu bude objasněn v dalších kapitolách.

Výstup kompilace projektu byl uložen na SD kartě v adresáři `/home/vehicle/`.

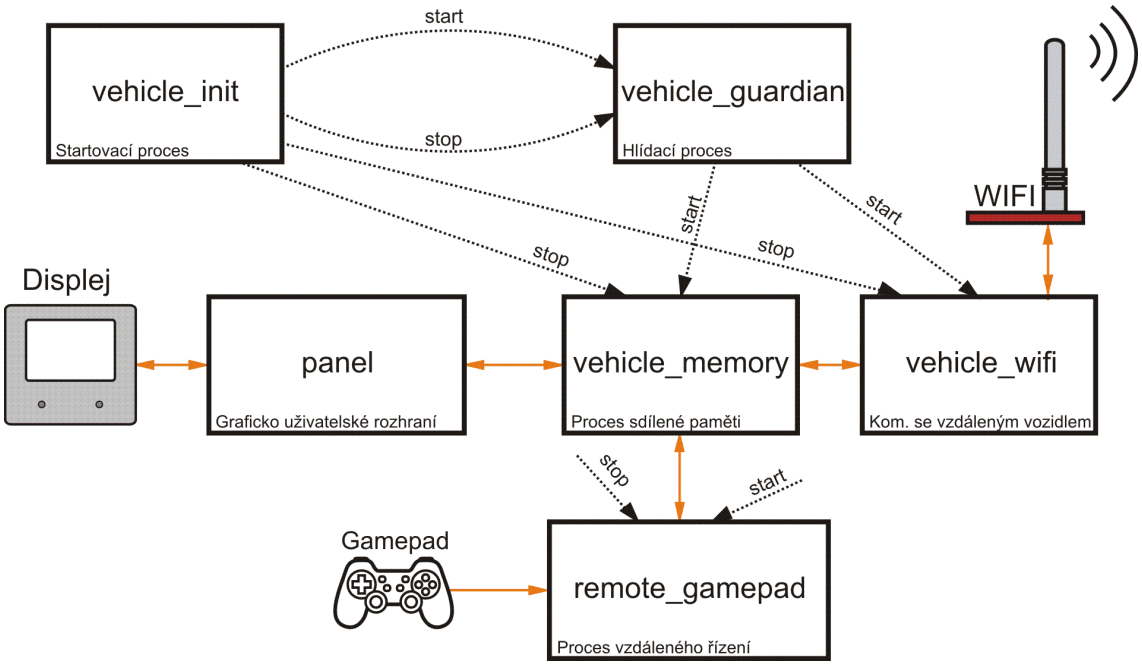
11 Realizace řídicích algoritmů

V první části budou objasněny funkce vytvořených procesů včetně vývojového diagramu hlavní aplikace panel. Poté bude objasněna jedna z důležitých funkcí panelu a na závěr budou popsány režimy řízení průzkumného vozidla přes ovládací panel.

11.1 Řídicí procesy

Výhody použití operačního systému byly částečně probrány v kapitole Operační systém. Nebylo zde ale zmíněno, že operační systém umožňuje programátorovi při tvorbě své aplikace rozdělit jeden proces do několika podprocesů (vláken) a tím zefektivnit svou práci. Jednotlivá vlákna či procesy vykonávají různě složité operace a nemusí byt mezi sebou nutně závislé, ale mohou spolu vzájemně komunikovat a předávat si zpracovaná data nebo výsledky. Komunikace mezi procesy může být realizována zasíláním zpráv, vytvořením routy, meziprocesní komunikací nebo využitím společné (sdílené) paměti.

Blokové schéma na obrázku pod textem znázorňuje použité procesy a vazby mezi nimi.



Obr. 45 Blokové schéma všech procesů

Některé procesy byly převzaty z průzkumného vozidla, v rámci této práce byly vytvořeny pouze procesy **panel** a **remote_gamepad**. Převzaté procesy budou popsány jenom stručně pro pochopení funkčnosti.

Proces **vehicle_init** sloužící pro nastartování ostatních procesů s možností vypnutí nebo jejich restartování. To se provede spuštěním procesu s parametrem (např. pro nápovědu: `--help`).

Dalším procesem je **vehicle_guardian**. Ten má za úkol hlídat ostatní procesy v případě selhání a jejich ukončení. Zajišťuje opětovné spuštění procesu. Před začátkem spuštění procesu je uloženo jeho PID číslo do souboru `/tmp/vehicle/jmeno_procesu.pid`, které guardian hlídá, a pokud se v tabulce všech procesů nevyskytuje, spustí proces znova.

Nejdůležitějším procesem je **vehicle_memory**. Jedná se o sdílenou paměť, ke které přistupují ostatní procesy. Se sdílenou pamětí vznikají požadavky na synchronizaci procesů v době zápisu a čtení. Toto je řešeno jednoduchým mutexem, který sdílenou paměť uzamkne po dobu její manipulace a tím je paměť chráněna, aby nedocházelo ke změně v době čtení. Účastník by tak získával neplatná nebo nesmyslná data.

Komunikaci zajišťuje proces **vehicle_wifi**. Stará se o přeposílání dat ze sdílené paměti do WiFi modulu. Komunikační protokol je rozsáhlá struktura, která obsahuje jak data pro vzdálený řídicí systém, tak pro řídicí jednotku a ostatní moduly na vozidle. Tudíž není nutné přeposílání celé struktury. Proces tedy odesílá pouze data určena pro řídicí jednotku na vozidle. Navíc je tento proces optimalizován tak, aby se přenášelo vždy podobné množství dat a tím mohlo dojít ke snížení přenosové rychlosti na sběrnici.

Proces **remote_gamepad** obstarává funkci manuálního řízení. Z předem definované struktury, která byla zmíněna v kapitole Joystick (Gamepad), získává informace o vzniklých událostech. Je-li tato událost definovaná, například stisk konkrétního tlačítka, zapíše se do sdílené paměti. Odeslání požadavku do vozidla už zajistí proces **vehicle_wifi**. Proces je nezávislý na grafické aplikaci, jelikož disponuje funkcí přepínání mezi manuálním a automatickým režimem. Protože každý joystick může mít jinou citlivost, je nutné jej kalibrovat. Kalibrace se prování automaticky po spuštění procesu podle hodnot, které byly už jednou nastaveny.

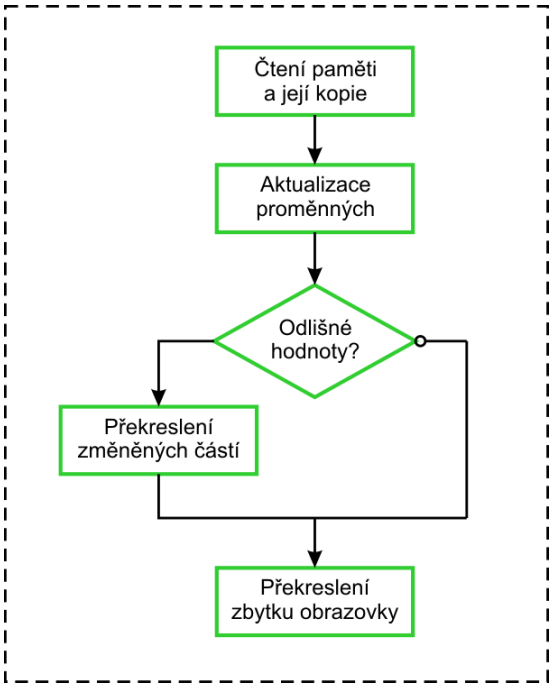
Proces nastavuje natočení kol v rozsahu (-60, 60), nastavení požadovaných konstantních otáček motoru (-3000, 7000), přepnutí mezi automatickým a ručním režimem, aktivaci brzdy, zapnutí nebo vypnutí světel a aktivaci klaksonu.

Neustále se prolínají pojmy joystick a gamepad. V podstatě jsou to podobná zařízení, kde jedno z nich má dvě analogové páky a více tlačítek. Z hlediska Linuxu se obě připojí jako stejné zařízení. Využívají stejné API, tudíž je i stejná struktura pro získávání dat. Dalo by se říci, že nezáleží na tom, jestli bude připojeno jedno nebo druhé. Z počátku byla aplikace vytvořena pro gamepad, protože je pro manuální řízení vozidla pohodlnější než joystick, ale z důvodu špatné citlivosti byl proces přizpůsoben pro joystick.

Co se týče **panelu** nejezná se zde o klasický proces ale o aplikaci, protože poskytuje uživateli grafické uživatelské rozhraní. Na rozdíl od ostatních procesů, je tato aplikace spuštěna pouze po startu operačního systému a není hlídaná guadianem. Aplikace je spuštěna přes skript, který připojí dotykovou obrazovku jako vstupní zařízení, a ta pak funguje jako ukazatel. Komunikuje podobně jako ostatní procesy s procesem sdílené paměti, ze které získává data a poté je zobrazuje přes prvky grafického rozhraní na obrazovce.

11.2 Panel

Panel je aplikace řízená událostmi. Nejčastější událostí, která byla volána, bylo přerušení od časovače (100 milisekundové intervaly). To mělo za následek spuštění funkce, která zajišťovala periodické překreslování uměle vytvořených grafických prvků (mapy, otáčkoměry, grafy). Jelikož se jednalo o dosti rozsáhlou funkci, bude nastíněn její algoritmus (Obr. 46).



Obr. 46 Algoritmus funkce myTimer100ms

Zjednodušený vývojový diagram začíná čtením dat ze sdílené paměti. Sdílená paměť je proces, který je chráněn zámkem proti přepsání v době čtení a naopak. Je-li proces zamknut, musí ostatní procesy čekat, až bude opět odemčen. Proto by neměla být sdílená paměť dlouho blokována, a tak je její obsah ihned zkopírován do lokální paměti pro pozdější použití.

Druhým bodem je aktualizace proměnných, tedy hodnot přijatých z průzkumného vozidla a jiných pomocných proměnných.

V textu o optimalizaci vykreslování bylo zmíněno, že není nutné překreslovat všechny objekty, ale pouze ty, u kterých došlo nějakým způsobem ke změně dat. Aktualizace textů na obrazovce není pro procesor tak náročná operace, jako překreslení celého obrázku. Proto se u náročnějších objektů rozlišuje, jestli je nutné je znovu překreslit, či nikoli.

Posledním krokem je překreslení zbytku grafického okna. Zde spadají objekty nejméně náročné na výpočetní výkon, u kterých není nutná optimalizace, nebo objekty, které není možné optimalizovat. Největší část grafických operací je prováděna právě zde. Patří sem vykreslování mapy, obraz z laserového senzoru nebo IR kamery.

11.3 Režimy řízení

Manuální režim slouží pro ruční ovládání vozidla. K tomuto účelu byl vytvořen samostatný proces **remote_gampad**. Co se týče řízení, tak ovládací panel umožňuje pouze přepínání mezi automatickým a manuálním režimem, a nastavení automatického režimu.

V automatickém režimu je vozidlo řízeno svými vlastními algoritmy. Tyto algoritmy jsou implementovány v jeho řídicí jednotce. Úkolem průzkumného vozidla v automatickém režimu je dojet na zadané souřadnice a vyhýbat se případným překážkám. V době vykonávání programu je řízení nezávislé na vzdáleném řídicím systému. Dalo by se říci, že v okamžiku, kdy jsou odeslány vstupní údaje do vozidla, a řízení je přepnuto do automatického módu, není panel potřeba.

Pro správnou funkci automatického režimu je nutné vybrat mapu, nastavit počáteční a cílovou pozici, zadat správné natočení vozidla v mapě a přepnout z manuálního režimu do automatického. Všechny tyto potřebné funkce byly vytvořeny a implementovány do ovládacího panelu a jsou k dispozici na jednom místě, na obrazovce map a navigace GPS (Obr. 38). Pokud vozidlo dosáhne cílové pozice, čeká na zadání nové cílové pozice. Ve skutečnosti je ale nutné zadat všechny parametry znova, protože řídicí jednotka vozidla musí znovu zpracovat údaje a přepočítat trajektorii pro splnění cíle. K tomuto účelu byla vytvořena resetovací sekvence (tlačítko reset na obrazovce), která dočasně nastaví všechny potřebné parametry do výchozího stavu, čeká na potvrzení ze strany vozidla a poté přepošle nové údaje.

Ovládací panel nerozlišuje, zda se jedná o ruční nebo automatické řízení, protože neustále vyhodnocuje a zobrazuje přijatá data. Historie jízdy vozidla nebo obraz ze senzorů se vykresluje v obou případech stejně.

12 Uložení vzdáleného řídicího systému

Poslední částí je vytvoření krabičky pro celou platformu. Krabička musí pojmout vývojovou sadu s rozšiřující deskou a procesorem, dotykový displej, WiFi modul a zdroj napájení.

Jako zdroj energie byl vybrán Lithiumpolymerový akumulátor s kapacitou 2200 mAh a napětím 11.1 V. Akumulátor by měl vydržet při průměrném odběru proudu 2.1 A s napětím 5 V kolem 2 hodin. Což je přibližně stejná doba jako výdrž průzkumného vozidla. Vývojová deska nedisponuje stabilizátorem napětí na 5 V, tudíž musel být doplněn.

Bylo vytvořeno schéma celé krabičky v AutoCADu. Krabička je složena ze dvou částí, které se k sobě na závěr přišroubují. Schéma nákresu je zobrazeno v příloze k nahlédnutí. Krabičky byla zkonstruována tak, aby vývojová deska, Wifi modul a akumulátor se stabilizátorem byly upevněny v jedné části a displej ke druhé části (víku). Aby nemusel být akumulátor v době nabíjení vytahován ven, je na boční straně krabičky vyveden konektor. Otvor je také na straně konektorů i.MX31.

Krabička byla vyrobena z plechu o šířce 0,8 mm. Na víku jsou umístěny dvě tlačítka (jedno pro zapnutí a druhé pro reset) a přepínač pro odpojení akumulátoru od i.MX31.Vývojovou desku lze tedy napájet jak z akumulátoru, tak ze standardního dodávaného zdroje. Hotová krabička je na obrázku (Obr. 47).

Obr. 47 Vzdálený řídicí systém s hotovou krabičkou

13 Závěr

Cílem diplomové práce bylo vytvořit vzdálený řídicí systém průzkumného vozidla, který by plnil funkci vizualizace a řízení. Na modelu průzkumného vozidla se vzdáleným řízením se podílela skupina čtyř diplomatů v rámci svých diplomových prací. Vozidlo bylo sestaveno ze skupiny modulů, které sbíraly důležité údaje ze svých senzorů a dále je poskytovaly vzdálenému systému. Uživatel obsluhující vzdálený řídicí systém měl pak přehled o všech získaných hodnotách a mohl vozidlo řídicí na dálku.

Diplomová práce v sobě zahrnovala výběr vhodného hardwaru a softwarových nástrojů. Jádrem celého systému byla vývojová deska i.MX31 s ARM architekturou. K tomuto systému byl připojen barevný LCD dotykový displej, který poskytoval rozhraní mezi uživatelem a grafickou aplikací. Jiný způsob řízení zajišťoval joystick, který musel být také připojen. Komunikaci mezi vzdáleným řízením a vozidlem probíhala bezdrátově prostřednictvím WiFi modulů, které byly připojeny k sériovému rozhraní. Moduly musely být nejprve nakonfigurovány. Byl použit šifrovaný přenos dat.

Do vývojové desky byl nasazen operační systém Timesys Linux, který byl zakoupen pro danou vývojovou sadou. Tento proces v sobě zahrnoval úpravu spouštěcího skriptu operačního systému, zkompilování jádra a poté vypálení do paměti. Do operačního systému musela být přidána podpora displeje a joysticku. Zde vznikaly problémy s dotykovou fólií, protože starší verze jádra operačního systému nejsou plně kompatibilní s použitým dotykovým displejem. Souborový systém byl uložen na externí paměťové SD kartě společně s aplikací.

Pro vývoj grafické aplikace musela být vybrána vhodná grafická knihovna. S ohledem na architekturu a možnosti vývojové desky byla zvolena Qt Embedded Linux grafická knihovna. Ta musela být nejprve zkompilována pro danou architekturu a poté implementována do souborového systému.

Ovládací panel, tedy hlavní řídicí aplikace, byl navržen tak, aby poskytoval uživateli přehledně všechny získané hodnoty ze senzorů průzkumného vozidla. Tyto hodnoty byly prezentovány různými grafickými prvky graficko-uživatelského rozhraní. Panel byl složen z několika obrazovek, kde každá z nich popisuje vždy část problému např. obrazovka senzorů pro detekci překážek, orientace v mapě nebo grafy naměřených hodnot. Byly vytvořeny nové grafické prvky. Některé grafické operace musely být optimalizovány, protože jejich výpočet zabíral velkou část výpočetního výkonu procesoru.

Byly vytvořeny řídicí procesy pro zajištění komunikace s průzkumným vozidlem, obsluhu sdílené paměti nebo pro manuální řízení. Některé z procesů byly implementovány i ve vozidle. Použití několika procesů umožnilo bezpečnější chod aplikace. Byla použita dvě vývojová prostředí, jedno pro vývoj grafické aplikace a druhé pro vývoj ostatních řídicích procesů.

V rámci této diplomové práci byla také navržena a sestavena krabička, do které byl uložen vzdálený řídicí systému s displejem, akumulátorem a WiFi modulem. Vzdálený řídicí systém je tedy přenosný a schopný běžet s baterií až 2 hodiny.

Byly splněny všechny stanovené cíle a body zadání. Diplomová práce společně s dokumentem v příloze slouží jako podrobný návod jak si připravit svou vývojovou stanici pro vývoj klasické nebo grafické aplikace na cílovém zařízení s ARM architekturou.

14 Použitá literatura

[1] *Timesys - Embedded Linux software* [online]. c2009 [cit. 2009-11-11]. Dostupný z WWW: <http://linuxlink.timesys.com/>.

[2] *Logic : A Product Development and Manufacturing Company* [online]. c2009 [cit. 2009-11-11]. Dostupný z WWW: <http://www.logicpd.com/>.

[3] *Freescale Semiconductors* [online]. c2009 [cit. 2009-11-11]. Dostupný z WWW: <http://www.freescale.com/>.

[4] SROVNAL, Vilém. *Operační systémy pro řízení v reálném čase.* Ostrava: VŠB-TU Ostrava, 2003. 212s

[5] *Qt for Embedded Linux* [online]. 2010 [cit. 2010-04-13]. Qt for Embedded Linux. Dostupné z WWW: < http://doc.qt.nokia.com/4.6/qt-embedded-linux.html>.

[6] *Wikipedia : the free encyclopedia* [online]. 2010, [cit. 2010-04-13]. X Window System. Dostupné z WWW: <http://en.wikipedia.org/wiki/X_Window_System>.

[7] *Wikipedia : the free encyclopedia* [online]. 2010, [cit. 2010-04-13]. Microwindows. Dostupné z WWW: <http://en.wikipedia.org/wiki/Microwindows>.

[8] *Wikipedia : the free encyclopedia* [online]. 2010, [cit. 2010-04-13]. Simple DirectMedia Layer. Dostupné z WWW: <http://en.wikipedia.org/wiki/Simple_DirectMedia_Layer>

[9] *OWSPA311g: Electrical and mechanical datasheet.* version 1.13. Sweden : [s.n.], 2008. 46s .

[10] *Wikipedia : the free encyclopedia* [online]. 2010, [cit. 2010-04-23]. Bezdrátová komunikace. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Bezdrátová_komunikace>.

15 Seznam příloh

Příloha I. Tabulka zpráv komunikačního protokolu
Příloha II. Schéma krabičky
Příloha III. Fotodokumentace
Příloha IV. Celkové blokové schéma (DVD – Přílohy/BlokoveSchema.pdf, 1xA2)
Příloha V. i.MX31 Litekit – Postup instalace a oživení
(DVD – / Přílohy /Imx31_v3.1.pdf, 48 stran)

Přiložené DVD dále obsahuje:

1. diplomova_prace.pdf – vlastní práce
2. diplomova_prace.doc – vlastní práce
3. bin/linux/filesystem.tar.gz – Souborový systém
4. bin/linux/vehicle.zip – Spustitelné soubory aplikace
5. bin/linux/zImage.zip – Spustitelné jádro linuxu
6. Dokumentace – Složka s dokumentacemi
7. Fotografie – Fotografie použité ve Fotodokumentaci
8. Fotografie/Aplikace – Fotografie z ovládacího panelu
9. Instalace/i.mx31 BSP – Instalační balíčky pro i.MX31 LiteKit
10. Instalace/Kernel – Jádro operačního systému
11. Instalace/Qt – Instalační balíčky Qt knihovny s vývojovým prostředím
12. Instalace/TimeStorm – Vývojové prostředí TimeStorm
13. Instalace/Tslib – Knihovny pro dotykový displej
14. Instalace/bootloader.tar – zavaděč Logic Loader
15. Instalace/ i.mx31-qt-ts-source.tar.gz – Zkompilovaná grafická knihovna s dotykovou obrazovkou, kterou stačí pouze nakopírovat do souborového systému
16. Instalace/Install.txt – Stručný návod pro přeložení a instalaci knihoven
17. Projekt – Složka se zdrojovými soubory grafické aplikace a ostatních procesů
18. Přílohy/Mapy.zip – Mapové podklady pro průzkumné vozidlo
19. Přílohy/Corel – Obrázky vytvořené v programu CorelDraw a použité v dokumentaci a programu
20. Přílohy/Krabička – Podklady pro krabičku vytvořené v programu AutoCAD

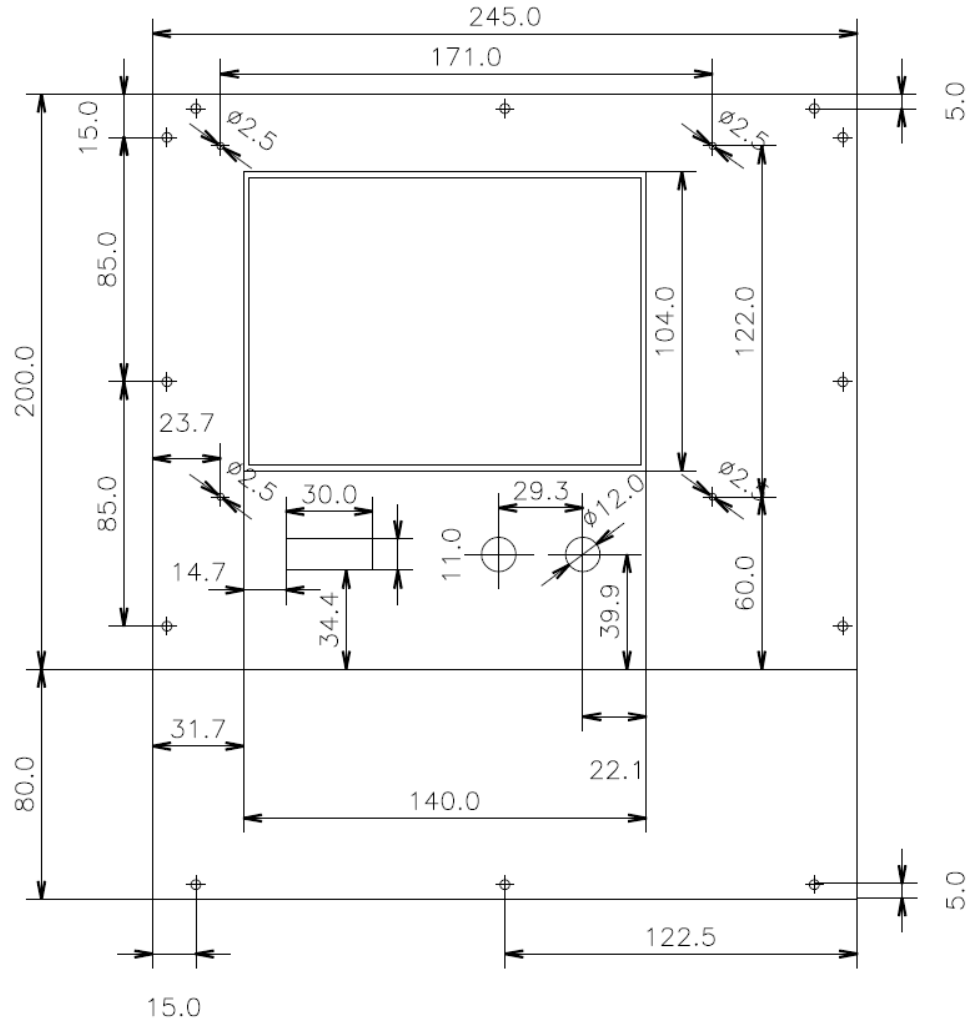
Příloha I Tabulka zpráv komunikačního protokolu

Název zprávy	Typ	Datový typ	Délka	Pod. j.	Vzd. p.	Jednotka	Popis
POLOHAX	0	Int32	5	S --> C	C --> P	[mm]	Absolutní X poloha v mapě
POLOHAY	1	Int32	5	S --> C	C --> P	[mm]	Absolutní Y poloha v mapě
NATOCENIVOZIDLA	2	Int16	3	S --> C	C --> P	[0.1°]	Natočení vozidla
PREDNIULTRAZVUK	3	Int16	3	S --> C	C --> P	[mm]	Pření ultrazvuk
ZADNIULTRAZVUK	4	Int16	3	S --> C	C --> P	[mm]	Zadní ultrazvuk
ANALOGOVEDALKOMERY	5	Int16	3	S --> C	C --> P	[mm]	4 analogové dálkoměry
GPSSOURADNICE	6	2*Int32	9	S --> C	C --> P	[0.000001°]	GPS souřadnice
GPSCAS	7	string[7]	8	S --> C	C --> P	[-]	GMT GPS čas
IRKAMERA	8	31*8*UInt8	249	S --> C	C --> P	[°C]	Teplota
TEPLOTA	9	UInt8	2	S --> C	C --> P	[°C]	Teplota okolí
AKCELEROMETRX	10	Int8	2	S --> C	C --> P	[-]	Údaj z akcelerometru X
AKCELEROMETRY	11	Int8	2	S --> C	C --> P	[-]	Údaj z akcelerometru Y
AKCELEROMETRZ	12	Int8	2	S --> C	C --> P	[-]	Údaj z akcelerometru Z
CIDLOOSVETLENI	13	Int8	2	S --> C	C --> P	[-]	Přední čidlo osvětlení
GPSSOURADNICE1	14	2*Int32	9	S <-- C	-	[0.000001°]	GPS souřadnice levého dolního rohu
GPSSOURADNICE2	15	2*Int32	9	-	-	[0.000001°]	GPS souřadnice pravého dolního rohu
NATOCENIKOL	16	Int8	2	S <-- C	C --> P	[°]	Natočení předních kol
AKTUALNIOTACKYMOTORU	17	Int16	3	S --> C	C --> P	[ot/min]	Aktuální otáčky motoru
PROUDMOTORU	18	Int16	3	S --> C	C --> P	[mA]	Proud motoru
BRZDA	19	Int8	2	S --> C	C --> P	[0,1]	Stav brzdy
MODBATERIE	20	Int8	2	S --> C	C --> P	[enum]	Nabíjení, slabá baterie, připraveno
STAVBATERIE	21	Int8	2	S --> C	C --> P	[%]	Stav baterie
AKTUALNISTAVMODULU	22	3*Int8	4	S --> C	C --> P	[enum]	Aktuální stav modulů
NABIJECIPROUD	23	Int16	3	S --> C	C --> P	[mA]	Nabíjecí proud
NABIJECINAPETI	24	Int16	3	S --> C	C --> P	[mV]	Nabíjecí napětí
ODEBIRANYPROUD	25	Int16	3	S --> C	C --> P	[mA]	Odebíraný proud
NAPETIBATERIE	26	Int16	3	S --> C	C --> P	[mV]	Napětí baterie
DOBADOVYBITI	27	Int16	3	S --> C	C --> P	[s]	Odhadovaný čas do vybití baterie
OTACKYMOTORU	28	Int16	3	S <-- C	-	[ot/min]	Požadované otáčky motoru
AKTIVACEBRZDY	30	Int8	2	S <-- C	-	[0,1]	Aktivace brzdy
STAVMODULU	31	3*Int8	4	-	-	[enum]	
RYCHLOSTVOZIDLA	32	Int32	5	-	C --> P	[0.01m/s]	Rychlost vozidla
VALUES	33	54*Int32	217	-	C --> P	[mm]	Změřené hodnoty z LMS 100 po 5°
VALUESX	34	54*Int32	217	-	-	[mm]	Změřené hodnoty z LMS 100 X
VALUESY	35	54*Int32	217	-	-	[mm]	Změřené hodnoty z LMS 100 Y
RUCNINATOCENIKOL	36	Int8	2	-	C <-- P	[°]	Ruční natočení kol
RUCNIOTACKYMOTORU	37	Int16	3	-	C <-- P	[ot/min]	Ruční nastavení otáček motoru
RUCNIAKTIVACEBRZDY	38	Int8	2	-	C <-- P	[0,1]	Ruční aktivace brzdy
AKTIVACERUCNIHOREZIMU	39	Int8	2	-	C <-- P	[0,1]	Aktivace ručního režimu

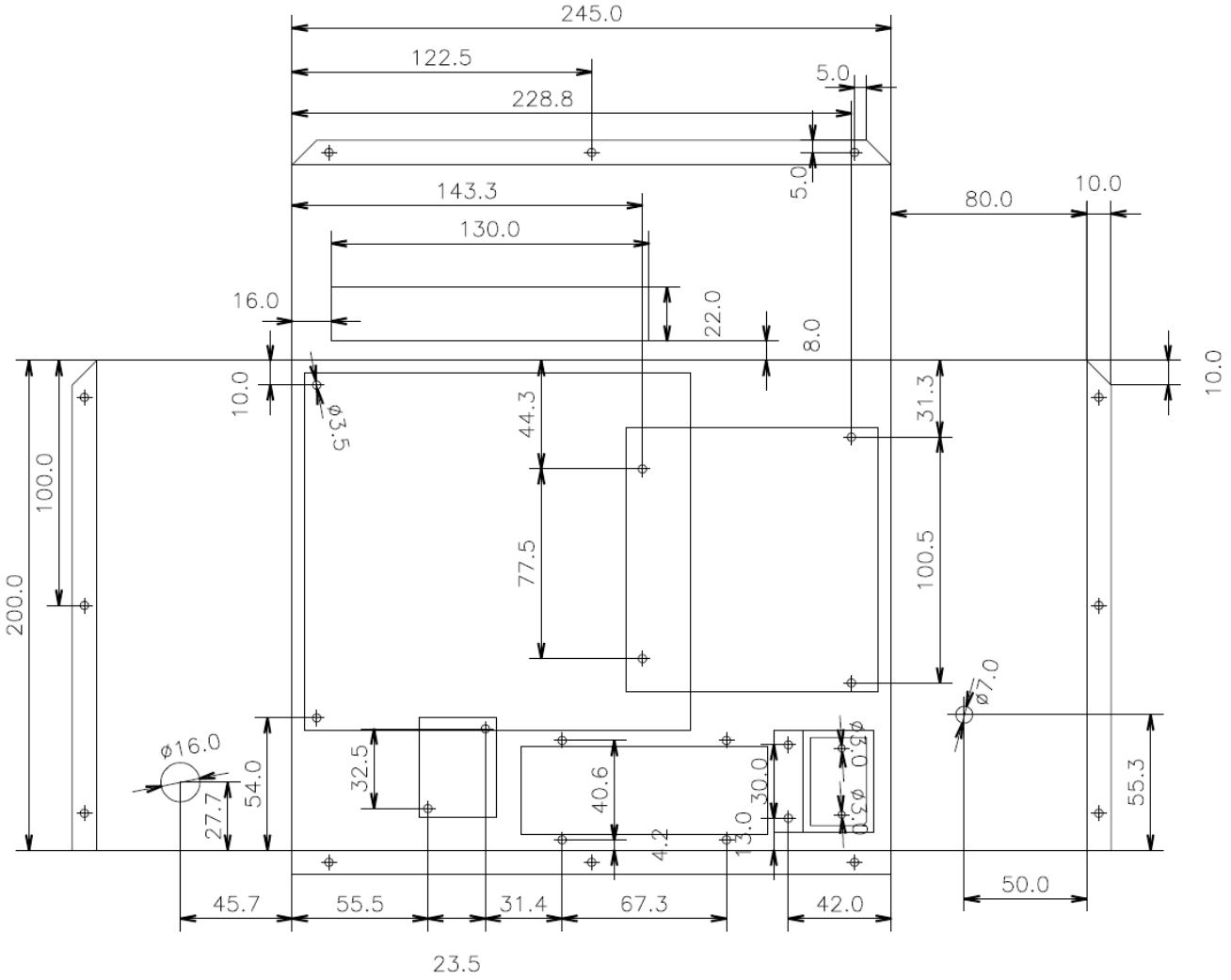
Název zprávy	Typ	Datový typ	Délka	Podříz. j.	Panel	Jednotka	Popis
VYCHOZIPOZICEX	40	Int32	5	S <-- C	-	[mm]	Výchozí pozice X
VYCHOZIPOZICEY	41	Int32	5	S <-- C	-	[mm]	Výchozí pozice Y
CILOVADESTINACEX	42	Int32	5	-	C <-- P	[mm]	Cílová destinace X
CILOVADESTINACEY	43	Int32	5	-	C <-- P	[mm]	Cílová destinace Y
TOTALSTOP	44	Int8	2	S <-- C	-	[0,1]	Nouzové zastavení
SENDISALIVE	45	Int8	2	S <-> C	C <-> P	[null]	Požadavek na stav komunikace
RECEIVEISALIVE	46	Int8	2	S <-> C	C <-> P	[null]	Odpověď na stav komunikace
STARTVYCHOZIPOZICEX	47	Int32	5	-	C <-- P	[mm]	Startovní pozice X
STARTVYCHOZIPOZICEY	48	Int32	5	-	C <-- P	[mm]	Startovní pozice Y
RUCNIZAPNUTISVETEL	49	UInt8	2	-	C <-- P	[0,1]	Ruční zapnutí světel
ZAPNUTISVETEL	50	UInt8	2	S <-- C	-	[0,1]	Požadavek na zapnutí světel
PLYN	51	UInt8	2	S --> C	C --> P	[-]	Čidlo plynů
TLAK	52	UInt16	3	S --> C	C --> P	[-]	Čidlo tlaku
CIDLOOSVETLENI2	53	Int8	2	S --> C	C --> P	[-]	Zadní čidlo osvětlení
ZKRAT1	54	UInt8	2	S --> C	C --> P	[0,1]	Zkrat na výstupech předního modulu
ZKRAT2	55	UInt8	2	S --> C	C --> P	[0,1]	Zkrat na výstupech zadního modulu
RUCNITOTALSTOP	56	UInt8	2	-	C <-- P	[0,1]	Ruční aktivace nouzového zastavení
VYCHOZINATOCENIVOZIDLA	57	Int16	3	S <-- C	C <-- P	[0.1°]	Výchozí natočení vozidla
KLAKSON	58	UInt8	2	S <-- C	-	[enum]	Zvuková signalizace
RUCNIKLAKSON	59	UInt8	2	-	C <-- P	[enum]	Zvuková signalizace

S - Podřízená jednotka
C - Řídicí jednotka vozidla
P - Vzdálený panel

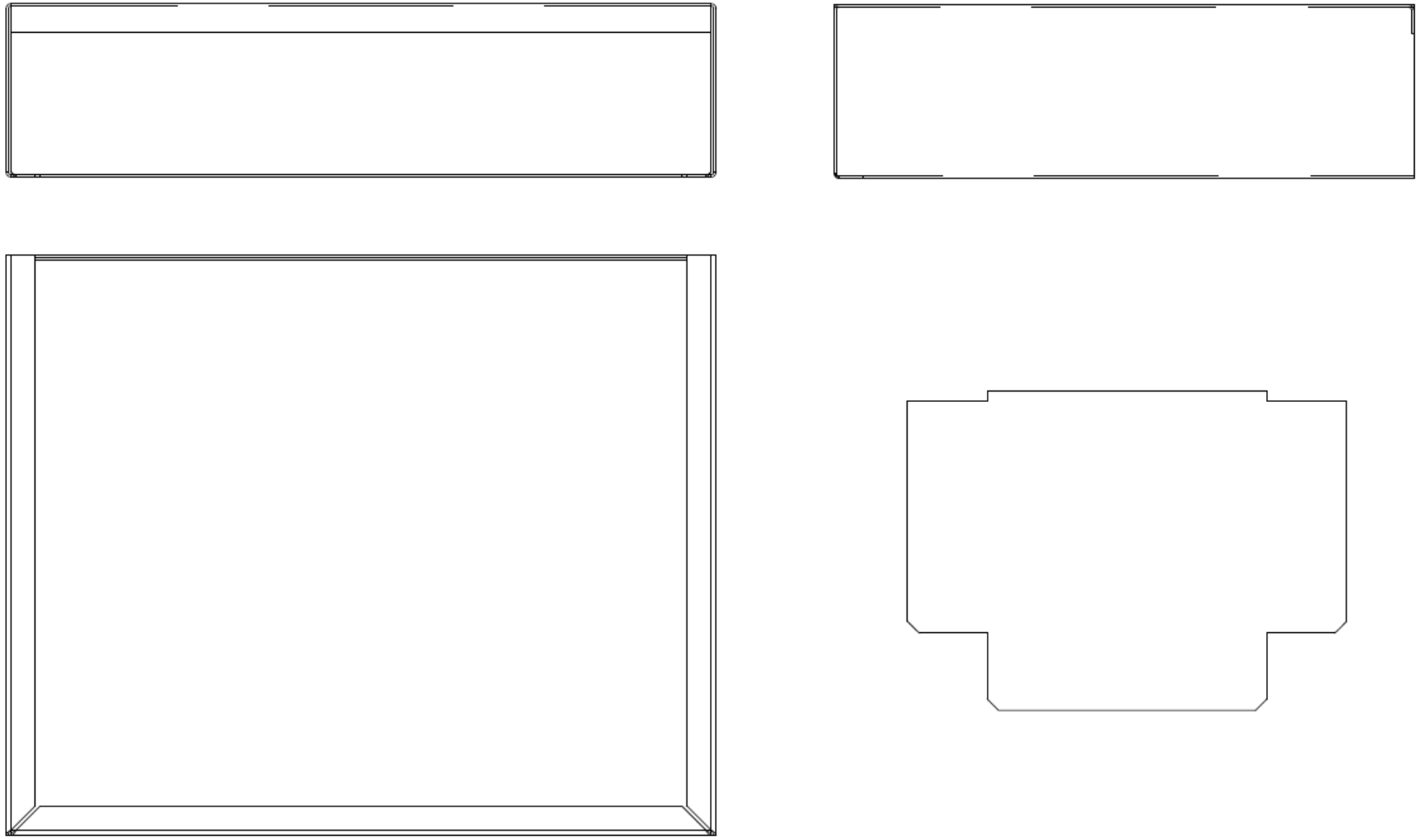
Příloha II Schéma krabičky



Horní část krabičky, kde je upevněn dotykový displej



Spodní část krabičky, kde je uchycen zbytek elektroniky s vývojovou deskou



Různé pohledy na krabičku včetně jejího rozložení

Příloha III Fotodokumentace

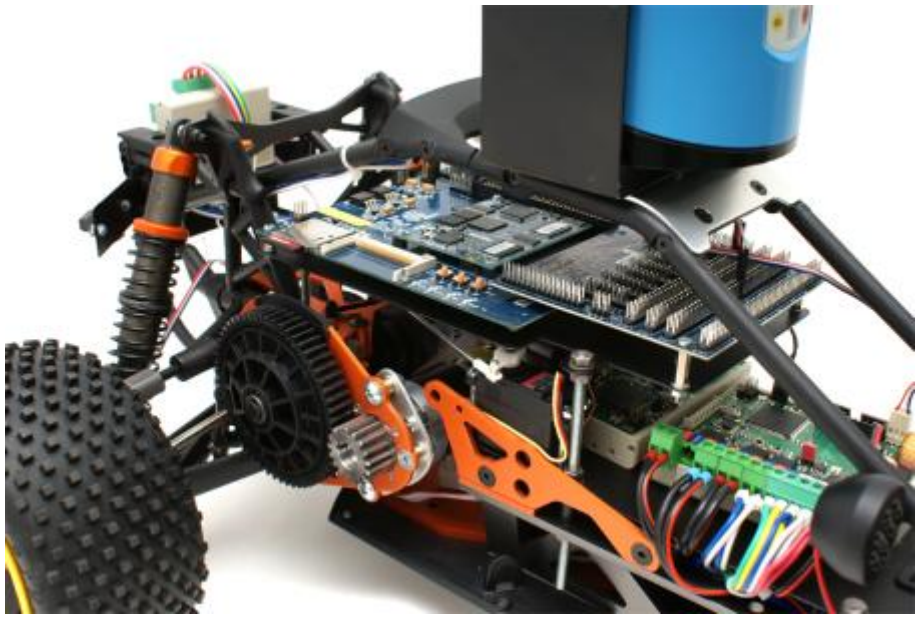
Displej vzdáleného ovládání s dotykovou fólií.

Rozšiřující deska pro vývojovou sadu i.MX31.

Baterie vzdáleného řídicího systému

Ukázka ovládacího panelu (bez krabičky)

Ukázka Ovládacího panelu a vozidla (bez krabičky)



Detailní pohled na pohon vozidla, správu napájení a řídicí jednotku (modrá deska)

Pohled na holé vozidlo s laserovým senzorem

Pohled zepředu na průzkumné vozidlo s namontovanou elektronikou

Průzkumné vozidlo po namontování řídicích modulů, řídicí jednotky a motoru.